Internalizing Presheaf Semantics: Charting the Design Space

Andreas Nuyts and Dominique Devriese
imec-DistriNet, KU Leuven, Belgium

May 14, 2018

Presheaf semantics [Hof97, HS97] are an excellent tool for modelling relational preservation properties of (dependent) type theory. They have been applied to parametricity (which is about preservation of relations) [AGJ14], univalent type theory (which is about preservation of equivalences) [BCH14, Hub15], and directed type theory (which is about preservation of morphisms) [RS17]. Of course after going through the endeavour of constructing a presheaf model of type theory, we want type-theoretic profit, i.e. we want internal operations that allow us to write cheap proofs of the 'free' theorems [Wad89] that follow from the preservation property concerned. There is currently no general theory of how we should craft such operations. Cohen et al. [CCHM16] introduced the final type extension operation[1] Glue, using which one can prove the univalence axiom and hence also the 'free' equivalence theorems it entails. In previous work with Vezzosi [NVD17], we showed that Glue and its dual, the initial type extension operation[1] Weld, can be used to internalize parametricity to some extent. Earlier, Bernardy et al. [BCM15, Mou16] had introduced completely different 'boundary-filling'[1] operations to internalize parametricity. Each of these operations has to our knowledge only been used in concrete models and hence their expressivity has not been compared. We have done some work to fill the hiatus: we consider and compare the various operators in more general presheaf categories. In this first step, we do not consider fibrancy requirements.

Universal extension operators The Glue and Weld types have meaning in any presheaf category [Nuy17] and do not by themselves internalize the specifics of the presheaf category. This makes them robust with respect to changes in the model, but also means that they can only be truly interesting in combination with other operations, such as the unweld operation below.

Given a type $\Gamma \vdash A$ type, an element of the subobject classifier $\Gamma \vdash P : \mathsf{Prop}$, a type $\Gamma, p : P \vdash T[p]$ type and a function $\Gamma, p : P \vdash f[p] : T[p] \to A$, the latter two defined only on the subobject $(\Gamma, p : P)$, the Glue operation yields the final extension (G, unglue) of $(T[p], f[p])$ to all of Γ,[2] where by extension we mean that both are equal when P holds. The Glue-type can be viewed as a record type with projections $\mathsf{unglue} : G \to A$, $\mathsf{red} : G \to (p : P) \to T[p]$ and $\mathsf{coh} : (g : G) \to (p : P) \to (f[p]\ (\mathsf{red}\ p\ g) =_A \mathsf{unglue}\ g)$, with the remarkable property that if P holds, then $G \equiv T[\_]$, $\mathsf{unglue} \equiv f[\_]$, $\mathsf{red}\ g\ \_ \equiv g$ and $\mathsf{coh}\ g\ \_ \equiv \mathsf{refl}$ definitionally. It is semantically a well-chosen pullback[3] of $A \xrightarrow{a \mapsto \lambda p.a} (P \to A) \xleftarrow{f \circ t \leftarrow\!\shortmid t} ((p : P) \to T[p])$.

The Weld operation takes the same input except that $\Gamma, p : P \vdash f[p] : A \to T[p]$ points the other way, and gives the initial extension (W, weld) of $(T[p], f[p])$. It can be seen as a quotient inductive type with constructors $\mathsf{weld} : A \to W$, $\mathsf{incl} : (p : P) \to T[p] \to W$ and $\mathsf{coh} : (p : P) \to (a : A) \to (\mathsf{incl}\ p\ (f[p]\ a) =_W \mathsf{weld}\ a)$, with the remarkable property that if P holds, then $W \equiv T[\_]$, $\mathsf{weld} \equiv f[\_]$, $\mathsf{incl}\ \_\ t \equiv t$ and $\mathsf{coh}\ \_\ a \equiv \mathsf{refl}$ definitionally. It is semantically a well-chosen pushout[3] of $A \xleftarrow{a \leftarrow\!\shortmid (p,a)} P \times A \xrightarrow{(p,a) \mapsto (p, f\ a)} (p : P) \times T$.

---

[1]Our terminology.

[2]Cohen et al. [CCHM16] allow the use of non-fibrant propositions P and therefore need to require f to be an equivalence so as to guarantee fibrancy of G, i.e. composition and transport. This obfuscates the finality property and makes the extension unique up to equivalence, although the Glue-extension is still final with respect to strict equality. When fibrancy is not a requirement, any f can be used and G is more clearly final. We emphasize that the presheaf construction of the Glue-type is identical in both contexts.

[3]Note that in a general presheaf category, there are no notions of homotopy (co)limits. By pullback/pushout, we mean the usual 1-categorical notions, defined up to isomorphism.

An additional eliminator for the Weld type internalizes aspects of the chosen presheaf model. Let the closed type $\mathbb{I}$ be semantically a representable object, i.e. a Yoneda-embedding of an object of the base category.[4] Assume that W was defined as above and that $\Gamma = (\Delta, i : \mathbb{I})$ and assume that $(\forall i.P) = \bot$. Then the existence of a function $\mathsf{unweld} : (\forall i.W) \to \forall i.A$ is sound, because elements of $\forall i.W$ necessarily come from the A-side of the pushout. A slightly more expressive operation which we called mill (after a milling cutter) exists and is of interest when $\bot \neq \forall i.P \neq \top$. Clearly, unweld is not sound when we replace $\mathbb{I}$ with an arbitrary type; thus, it internalizes something interesting about the model.

Cohen et al. [CCHM16] have only Glue and make this operation interesting by providing an operation for composition and transport. Our earlier work [NVD17] has Glue and Weld but not unweld/mill (for the sole reason that unweld/mill had not been conceived yet); there we make these operations interesting by internalizing the identity extension lemma of parametricity [Rey83].

Boundary-filling operators Every representable object $\mathbb{I}$ has a largest subobject $\partial\mathbb{I} \subsetneq \mathbb{I}$ which we call its boundary; for interval types this typically contains just the endpoints. Bernardy et al.'s operators [BCM15] allow us to extend data defined on $(\Gamma, i : \partial\mathbb{I})$ to $(\Gamma, i : \mathbb{I})$; we call the diff of this (i.e. the information that is forgotten when we restrict to the boundary) a *filler*. There is an operator Φ for functions, which allows us to extend $\Gamma, i : \partial\mathbb{I} \vdash f[i] : (a : A[i]) \to B[i, a]$ to all of $\mathbb{I}$ by giving an extension of $\Gamma, \alpha : (i : \mathbb{I}) \to A[i], i : \partial\mathbb{I} \vdash f[i]\ (\alpha\ i) : B[i, \alpha\ i]$, i.e. by giving an action on fillers. Note that this internalizes the interpretation of the function type in presheaf semantics and also corresponds closely to Reynolds' relational interpretation of the function type [Rey83]. The operator Ψ for types allows us to extend $\Gamma, i : \partial\mathbb{I} \vdash T[i] : \mathcal{U}$ to all of $\mathbb{I}$ by giving a (proof-relevant) predicate $P : ((i : \partial\mathbb{I}) \to T[i]) \to \mathcal{U}$. If $\partial\mathbb{I}$ contains just two endpoints, then P is simply a binary relation. Fillers of terms $t : (i : \partial\mathbb{I}) \to T[i]$ to $(i : \mathbb{I}) \to \Psi\ (j.T[j])\ P\ i$ are then in correspondence with elements of $P\ t$.

These operators are formulated in terms of representable objects and boundaries, which to a large extent reveal the structure of the base category, making them more expressive than universal extension operations, but also more brittle with respect to changes in the model. Unfortunately, Φ and Ψ are at odds with the contraction rule for $\mathbb{I}$. Indeed, repeated use of Φ allows us to define a square of functions $i, j : \mathbb{I} \vdash f[i, j] : A[i, j] \to B[i, j]$ by giving the four endpoints $f[0, 0], f[0, 1], f[1, 0], f[1, 1]$; specifying how to act on lines of type $(i : \mathbb{I}) \to A[i, 0], (i : \mathbb{I}) \to A[i, 1], (j : \mathbb{I}) \to A[0, j], (j : \mathbb{I}) \to A[1, j]$, as well as on squares $(i, j : \mathbb{I}) \to A[i, j]$. Then contraction would allow us to form $k : \mathbb{I} \vdash f[k, k] : A[k, k] \to B[k, k]$, but we have not specified how $f[k, k]$ should act on a diagonal of type $(k : \mathbb{I}) \to A[k, k]$.

The solution is to disallow simultaneous substitutions in which the object substituted for $i : \mathbb{I}$ shares any dependencies with variables on its left [BCM15, Mou16]. This can be formalized semantically using a separated product [Pit13, BCH14], a concept that is unfortunately not defined in every presheaf category.

Similarly, Φ is incompatible with connections $\vee, \wedge : \mathbb{I}^2 \to \mathbb{I}$ as used by Cohen et al. [CCHM16]. The Ψ-operation is compatible, but becomes underspecified in absence of Φ. We may then consider an initial and a final Ψ-operation; isomorphism of these two types is implied by, but also seems to imply the existence of Φ.

Results We proved a few theorems that clarify the expressivity of the aforementioned operators. First, Glue, Weld and unweld/mill can be implemented using Ψ, Φ and meta-theoretic induction, provided that the base category is in some sense well-founded. Secondly, the initial Ψ-operation can be implemented using Weld, mill and a 'freshness' predicate that arises by viewing the separated product as a subobject of the cartesian product. Thirdly, Φ *cannot* be implemented even when Glue, Weld, unweld/mill and the freshness predicate are all available. Fourthly, parametricity of a predicative System F universe defined in MLTT, can be proven using Ψ and Φ and an interval with two endpoints, but a lemma that we likely want to rely on, cannot be proven using Glue, Weld, unweld/mill, the freshness predicate and the same interval.

Acknowledgements *Andreas Nuyts and Dominique Devriese hold a Ph.D. Fellowship and a Postdoctoral Mandate (resp.) from the Research Foundation - Flanders (FWO).*

[4] In all applications that we are aware of, the only such types ever internalized are the relational/homotopy interval and, less interestingly, the unit type.

References

[AGJ14] Robert Atkey, Neil Ghani, and Patricia Johann. A relationally parametric model of dependent type theory. In *Principles of Programming Languages*, 2014. doi:10.1145/2535838.2535852.

[BCH14] Marc Bezem, Thierry Coquand, and Simon Huber. A Model of Type Theory in Cubical Sets. In *19th International Conference on Types for Proofs and Programs (TYPES 2013)*, volume 26, pages 107–128, Dagstuhl, Germany, 2014. URL: http://drops.dagstuhl.de/opus/volltexte/2014/4628, doi:10.4230/LIPIcs.TYPES.2013.107.

[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.

[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.

[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.

[Hub15] Simon Huber. A model of type theory in cubical sets. Licentiate's thesis, University of Gothenburg, Sweden, 2015.

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

[Nuy17] Andreas Nuyts. A model of parametric dependent type theory in bridge/path cubical sets. Technical report, KU Leuven, Belgium, 2017. URL: https://arxiv.org/abs/1706.04383.

[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.

[Pit13] Andrew M. Pitts. *Nominal Sets: Names and Symmetry in Computer Science*. Cambridge University Press, New York, NY, USA, 2013.

[Rey83] John C. Reynolds. Types, abstraction and parametric polymorphism. In *IFIP Congress*, pages 513–523, 1983.

[RS17] E. Riehl and M. Shulman. A type theory for synthetic ∞-categories. *ArXiv e-prints*, May 2017. arXiv:1705.07442.

[Wad89] Philip Wadler. Theorems for free! In *FPCA '89*, pages 347–359, New York, NY, USA, 1989. ACM. doi:10.1145/99370.99404.