Presheaves on Purpose

Conor McBride[1,2]

[1] University of Strathclyde
conor.mcbride@strath.ac.uk
[2] Quantinuum

Abstract

In current dependent type theories, we give types to the indices of inductive datatypes but we say very little about the structure of those index types. Categorically, they are treated as discrete, as the only structure automatically respected is equality. Here, I give a universe construction for datatypes indexed over small categories which are functorial by construction, hence presheaves, with a definition and proof given once for all.

1 Introduction

Whenever we find ourselves engineering coincidences, something is wrong. If we cannot articulate the design choices whose consequences we propagate, something is wrong. Let me show you something wrong. In Haskell, Agda, Idris, Lean or Coq (to name but a handful), I can write a function (pronounced ‘thin’)

$$\frac{t \;:\; \mathsf{Term}\,n \quad \theta \;:\; n \sqsubseteq m}{t\!\uparrow\!\theta \;:\; \mathsf{Term}\,m} \qquad\qquad t\!\uparrow\!\iota = t \qquad\qquad t\!\uparrow\!(\theta \mathbin{⨾} \phi) = (t\!\uparrow\!\theta)\!\uparrow\!\phi$$

where $\mathsf{Term}\,n$ is the type of *well scoped* lambda-terms with n free variables in scope, and $n \sqsubseteq m$ is the type of order-preserving injections—*thinnings*—embedding n free variables into a larger scope with m free variables[1]. Moreover, with synthetic astonishment I can subsequently prove that $\cdot\!\uparrow\!\cdot$ respects thinning identity, ι, and composition, $⨾$. In other words, I extend Term to a *presheaf* over (op-)thinnings — a functor into Type. And what is wrong is that *I* do it *myself*.

2 Worked Example

Let us take a closer look at the constructors of $\mathsf{Term}\,\cdot$ and the action of $\cdot\!\uparrow\!\cdot$:

$$\frac{x \;:\; 1 \sqsubseteq n}{\mathsf{var}\;x \;:\; \mathsf{Term}\,n} \qquad \frac{f, s \;:\; \mathsf{Term}\,n}{\mathsf{app}\;f\;s \;:\; \mathsf{Term}\,n} \qquad \frac{t \;:\; \mathsf{Term}\,(n{+}1)}{\mathsf{lam}\;t \;:\; \mathsf{Term}\,n}$$

$$(\mathsf{var}\;x)\!\uparrow\!\theta = \mathsf{var}\;(x \mathbin{⨾} \theta) \quad\Big|\quad (\mathsf{app}\;f\;s)\!\uparrow\!\theta = \mathsf{app}\;(f\!\uparrow\!\theta)\;(s\!\uparrow\!\theta) \quad\Big|\quad (\mathsf{lam}\;t)\!\uparrow = \mathsf{lam}\;(t\!\uparrow\!(\theta{+}1))$$

Note that for var, θ acts by *postcomposition*. Moreover, for lam, the postfix $\cdot{+}1$ which happens to n in the type looks a lot like the $\cdot{+}1$ which happens to θ in the function. What is the latter? Thinnings $n \sqsubseteq m$ (determining particular choices of n things from m) are generated thus:

$$\frac{}{0 \;:\; 0 \sqsubseteq 0} \qquad \frac{\theta \;:\; n \sqsubseteq m}{\theta{\curlywedge} \;:\; n \sqsubseteq m{+}1} \qquad \frac{\theta \;:\; n \sqsubseteq m}{\theta{+}1 \;:\; n{+}1 \sqsubseteq m{+}1}$$

---

[1] I like m to be larger than n. Count the sticks.

The $\cdot\!+\!1$ action on a thinning coincides with the $\cdot\!+\!1$ action on source and target scopes. My overloading of the constructors is deliberate emphasis. The definitions of identity and composition confirm that $\cdot\!+\!1$ is a *functor*.

$$
\begin{array}{rcl@{\qquad\qquad}rcl}
\iota_0 & = & 0 & 0 \mathbin{;} 0 & = & 0 \\
 & & & \theta \mathbin{;} (\phi\curlywedge) & = & (\theta \mathbin{;} \phi)\curlywedge \\
 & & & (\theta\curlywedge) \mathbin{;} (\phi{+}1) & = & (\theta \mathbin{;} \phi)\curlywedge \\
\iota_{(n+1)} & = & \iota_n{+}1 & (\theta{+}1) \mathbin{;} (\phi{+}1) & = & (\theta \mathbin{;} \phi){+}1
\end{array}
$$

In this talk, we shall learn to spot such structure and make presheaves on purpose. We should be able to express the definition of Term in a way that points out how $1 \sqsubseteq \cdot$ is a functor from Thin to Type and $\cdot\!+\!1$ from Thin to Thin, recovering the action of Thin on Term automatically.

3 Prospectus

My specific plan is to construct a universe of datatype descriptions, as in previous work [4], by syntactifying a class of strictly positive functors whose least fixpoint may then be taken. But where we previously described indexed containers [2], we may now consider strictly positive functors between presheaves.

This construction necessarily relies on some formalisation of category theory, and it is a local non-goal for this to be a comprehensive treatment. We can get a long way with a notion of *small* categories and functors between them. The thinnings are exactly such a category, with $\cdot\!+\!1$ exactly such a functor.

We may then, separately, say what it is to be a presheaf — a functor from a small 'index' category into Type, and lift type constructors accordingly. In particular, the covariant Hom-functor (arrows from a given source) gives such a presheaf, with $1 \sqsubseteq \cdot$ a case in point. With this notion in place, we can give a syntax of descriptions for functors between presheaves, including the identity, constants, pairing, composition with a small functor, and pointwise Π and Σ over sets. When the source and target index categories of a description coincide, we may take a least fixpoint. We show once, for all such descriptions that this fixpoint a presheaf itself, obtaining the action of arrows in the index category and the proofs that identity and composition are respected.

4 Future Work

The idea that our indexed datatypes should respect more than discrete structure on indices is one small part of a broader agenda towards directed type theory [5, 3], but we can have it in our hands, now. Of course, we should very much like functor laws to hold *definitionally*, and that should be workable [1]. Let us see how much more categorical structure we can build into type theory for the price of having enough language to point it out.

References

[1] Guillaume Allais, Conor McBride, and Pierre Boutillier. New equations for neutral terms: a sound and complete decision procedure, formalized. In Stephanie Weirich, editor, *Proceedings of the 2013 ACM SIGPLAN workshop on Dependently-typed programming, DTP@ICFP 2013, Boston, Massachusetts, USA, September 24, 2013*, pages 13–24. ACM, 2013.

[2] Thorsten Altenkirch, Neil Ghani, Peter G. Hancock, Conor McBride, and Peter Morris. Indexed containers. *J. Funct. Program.*, 25, 2015.

[3] Thorsten Altenkirch and Jacob Neumann. Synthetic 1-categories in directed type theory. *CoRR*, abs/2410.19520, 2024.

[4] James Chapman, Pierre-Évariste Dagand, Conor McBride, and Peter Morris. The gentle art of levitation. In Paul Hudak and Stephanie Weirich, editors, *Proceeding of the 15th ACM SIGPLAN international conference on Functional programming, ICFP 2010, Baltimore, Maryland, USA, September 27-29, 2010*, pages 3–14. ACM, 2010.

[5] Daniel R. Licata and Robert Harper. 2-dimensional directed type theory. In Michael W. Mislove and Joël Ouaknine, editors, *Twenty-seventh Conference on the Mathematical Foundations of Programming Semantics, MFPS 2011, Pittsburgh, PA, USA, May 25-28, 2011*, volume 276 of *Electronic Notes in Theoretical Computer Science*, pages 263–289. Elsevier, 2011.