Coherences for the Container Model of Type Theory

Stefania Damato and Thorsten Altenkirch

University of Nottingham, Nottingham, UK
{stefania.damato, thorsten.altenkirch}@nottingham.ac.uk

Several models of type theory given as categories with families (CwFs), including the set theoretic model and the presheaf model, implicitly assume uniqueness of identity proofs (UIP) in order to precisely fit the definition of a CwF. Our goal is to construct such a model using containers, while working in Martin-Löf type theory without assuming UIP. In their abstract [5], Altenkirch and Kaposi propose a container model of type theory as a CwF, which however raises some coherence issues in the absence of UIP that need to be addressed in order to complete the model. In this talk, we present ongoing work addressing these issues.

Motivation The theory of containers (a.k.a. polynomial functors) has proved very useful in providing semantics for inductive types and inductive families [1, 4]. Our goal is to extend this theory to provide semantics for more general classes of inductive types that are less well understood, specifically inductive-inductive types (IITs) and quotient inductive-inductive types (QIITs). Constructing this model is a prerequisite to this approach. In our proposed (Q)IIT semantics [3, 2], a data type constructor is represented by a pair of functors which have to be expressable as container functors. Therefore, we need a general way to express any type context as a container. This can be achieved by constructing a container model of type theory.

The container model We briefly outline the container model using CwFs as our notion of a model of type theory, ignoring coherence issues for now. This model can be seen as a restriction of the presheaf model, where we only consider endofunctors on **Set** that are container functors.

- The base category is the category of set-containers **Cont**. The objects, corresponding to contexts, are set-containers, i.e. pairs $S_\Gamma : \mathsf{Set}, P_\Gamma : S_\Gamma \to \mathsf{Set}$[1] written as $S_\Gamma \lhd P_\Gamma$, and morphisms $\sigma : S_\Gamma \lhd P_\Gamma \to S_\Delta \lhd P_\Delta$, corresponding to substitutions, are container morphisms, i.e. pairs $\sigma_s : S_\Gamma \to S_\Delta, \sigma_p : (s : S_\Gamma) \to P_\Delta\,(\sigma_s\,s) \to P_\Gamma\,s$. The terminal object, corresponding to the empty context, is $\mathbf{1} \lhd \mathbf{0}$. Every container $S_\Gamma \lhd P_\Gamma$ gives rise to an endofunctor $[\![S_\Gamma \lhd P_\Gamma]\!] : \mathbf{Set} \to \mathbf{Set}$, defined on objects by $[\![S_\Gamma \lhd P_\Gamma]\!]\,X \coloneqq \sum(s : S_\Gamma)(P_\Gamma\,s \to X)$, and every container morphism gives rise to a natural transformation between such endofunctors.
- A type over context Γ is given by a generalised container A, i.e. a pair $S_A : \mathsf{Set}$, $P_A : S_A \to |\int [\![\Gamma]\!]|$, which gives rise to a functor $[\![S_A \lhd P_A]\!] : \int [\![\Gamma]\!] \to \mathbf{Set}$, defined on objects by $[\![S_A \lhd P_A]\!]\,\gamma \coloneqq \sum(s : S_A)(\int [\![\Gamma]\!](P_A\,s, \gamma))$.[2] Given A and $\sigma : \Delta \to \Gamma$, type sub-

[1] Set refers to the universe of h-sets.

[2] $|\int [\![\Gamma]\!]|$ denotes the type of objects of the category of elements of $[\![\Gamma]\!]$. P_A has the following components: $P_A^X : S_A \to \mathsf{Set}$, $P_A^s : S_A \to S_\Gamma$, $P_A^f : (s : S_A) \to P_\Gamma(P_A^s\,s) \to P_A^X\,s$.

stitution $A[\sigma]$ is a generalised container $S_{A[\sigma]} \triangleleft P_{A[\sigma]}$ over $\int [\![\Delta]\!]$, defined using pullback and pushout respectively as follows. In the rightmost diagram, assume $s : S_{A[\sigma]}$, and that we can transport along the equation given by the leftmost diagram.

$$\begin{array}{ccc} S_{A[\sigma]} & \xrightarrow{\text{snd}} & S_A \\ {\scriptstyle \text{fst}}\downarrow & \lrcorner & \downarrow{\scriptstyle P_A^s} \\ S_\Delta & \xrightarrow[\sigma_s]{} & S_\Gamma \end{array} \qquad \begin{array}{ccc} P_\Gamma(\sigma_s(\text{fst}\, s)) & \xrightarrow{P_A^f(\text{snd}\, s)} & P_A^X(\text{snd}\, s) \\ {\scriptstyle \sigma_p(\text{fst}\, s)}\downarrow & & \downarrow{\scriptstyle \text{inr}} \\ P_\Delta(\text{fst}\, s) & \xrightarrow[\text{inl}]{} & P_{A[\sigma]}^X\, s \end{array}$$

The $P^s_{A[\sigma]}$ and $P^f_{A[\sigma]}$ components of $P_A[\sigma]$ are then defined as fst and inl respectively.

- A term of type A in context Γ is given by a 'dependent natural transformation' a from $[\![\Gamma]\!]$ to $[\![A]\!]$, which we denote as $a : \int_{X:\mathsf{Set}}(\gamma : [\![\Gamma]\!]X) \to [\![A]\!](X, \gamma)$. Given a and $\sigma \colon \Delta \to \Gamma$, term substitution $a[\sigma]$ is roughly the natural transformation $a \circ [\![\sigma]\!]$.
- Given context Γ and type A over Γ, the extended context $\Gamma.A$ is defined as the set-container $S_A \triangleleft P_A^X$.

Coherence issues One coherence issue arising in the proposed model is that we have a *groupoid* of types (and also of contexts), whereas the definition of a CwF requires this to be an h-set. This being a groupoid means that the functor laws of the functor Ty interpreting types, namely the laws $A[\text{id}] = A$ and $A[\delta \circ \sigma] = A[\delta][\sigma]$, hold up to higher equalities, which would also need to be checked. Another coherence issue is that in our model, these same functor laws do not hold strictly but only up to isomorphism, due to our definition of type substitution using pullback and pushout, which are only unique up to isomorphism.

There are two alternative ways we can solve these issues. One way is to generalise the definition of a CwF so that types (and contexts) can be groupoids (or higher types), a so-called 'higher' or 'coherent' CwF. This would require interpreting the syntax of type theory, which is the initial set-CwF, as a higher CwF. More specifically, we would need to prove that the initial coherent CwF is a set-CwF. Another way is to strictify our proposed model so that it fits the regular definition of a CwF. We are currently taking the latter approach. The first coherence issue can be tackled by using an inductive-recursive universe $U : \mathsf{Set}, El \colon\ U \to \mathsf{Set}$, such that the elements of U are codes for our actual types, which are decoded by El. The second coherence issue can be tackled by redefining the functor Ty in a coherent way by accumulating compositions. We define $Ty'\,\Gamma \coloneqq \sum(\sigma \colon \Gamma \to \Delta)(Ty\ \Delta)$ for some context Δ, which satisfies the required functor laws strictly.

Future work This model is a starting point for our actual application of (Q)IIT semantics, which requires the model to be constructed with respect to a fixed category of algebras. This means that the base category needs to be the category of generalised containers instead of set-containers. We aim to construct this generalised model in the future.

References

[1] Michael Abbott, Thorsten Altenkirch, and Neil Ghani. Containers: Constructing strictly positive types. *Theoretical Computer Science*, 342(1):3–27, 2005.

[2] Thorsten Altenkirch, Paolo Capriotti, Gabe Dijkstra, Nicolai Kraus, and Fredrik Nordvall Forsberg. Quotient inductive-inductive types. In C. Baier and U. Dal Lago, editors, *FoSSACS*, pages 293–310. Springer, 2018.

[3] Thorsten Altenkirch and Stefania Damato. Specifying QIITs using Containers, 2023. Talk abstract at HoTT/UF, available at https://stefaniatadama.com/talks/abstract_hott_uf_2023.pdf.

[4] Thorsten Altenkirch, Neil Ghani, Peter Hancock, Conor McBride, and Peter Morris. Indexed containers. *Journal of Functional Programming*, 25:e5, 2015.

[5] Thorsten Altenkirch and Ambrus Kaposi. A container model of type theory, 2021. Talk abstract at TYPES, available at https://types21.liacs.nl/download/a-container-model-of-type-theory.