Coherences for the Container Model of Type Theory

Stefania Damato Thorsten Altenkirch

University of Nottingham, UK

Workshop on
Homotopy Type Theory/Univalent Foundations

3rd April 2024

Why do we need a container model of type theory?

For categorical semantics of (Q)IITs using containers!

Why do we need a container model of type theory?

For categorical semantics of (Q)IITs using containers!

Example 1: (Simplified) intrinsic syntax of type theory

```
data Con : Set
data Ty : Con → Set

data Con where
  ◇ : Con
  _,_ : (Γ : Con) (A : Ty Γ) → Con
  eq : (Γ : Con) (A : Ty Γ) (B : Ty (Γ , A)) →
     ((Γ , A) , B) ≡ (Γ , Σ Γ A B)

data Ty where
  ι : (Γ : Con) → Ty Γ
  Σ : (Γ : Con) (A : Ty Γ) → Ty (Γ , A) → Ty Γ
```

Why do we need a container model of type theory?

For categorical semantics of (Q)IITs using containers!

Constructors are expressed via argument and target functors [Altenkirch et al., 2018].

E.g. For `_,_ : (Γ : Con) (A : Ty Γ) → Con` we have:

$$L : \mathbf{A_1} \to \mathbf{Set}$$
$$L(C, T, e) :\equiv \textstyle\sum(\Gamma : C)(T\,\Gamma)$$

$$R : \textstyle\int L \to \mathbf{Set}$$
$$= (C : \mathrm{Set})(T : C \to \mathrm{Set})(e : C)(L(C, T, e)) \to \mathbf{Set}$$
$$R(C, T, e, \Gamma, A) :\equiv C$$

Why do we need a container model of type theory?

For categorical semantics of (Q)IITs using containers!

Constructors are expressed via argument and target functors [Altenkirch et al., 2018].

Conjecture:
Restricting ourselves to **container functors** ensures we can only construct **strictly positive** (Q)IITs.

Prerequisite:
A general way to **express type contexts as containers**, i.e. a container model of type theory.

The container model (outlined in [Altenkirch and Kaposi, 2021])

- Contexts are set-containers $S_\Gamma : \mathrm{Set} \lhd P_\Gamma : S_\Gamma \to \mathrm{Set}$ with extension functor

$$[\![S_\Gamma \lhd P_\Gamma]\!] : \mathbf{Set} \to \mathbf{Set}$$

The container model (outlined in [Altenkirch and Kaposi, 2021])

- Contexts are set-containers $S_\Gamma : \mathrm{Set} \triangleleft P_\Gamma : S_\Gamma \to \mathrm{Set}$ with extension functor

$$[\![S_\Gamma \triangleleft P_\Gamma]\!] : \mathbf{Set} \to \mathbf{Set}$$

- Substitutions are container morphisms

The container model (outlined in [Altenkirch and Kaposi, 2021])

- <u>Contexts</u> are set-containers $S_\Gamma : \mathrm{Set} \triangleleft P_\Gamma : S_\Gamma \to \mathrm{Set}$ with extension functor

$$[\![S_\Gamma \triangleleft P_\Gamma]\!] : \mathbf{Set} \to \mathbf{Set}$$

- <u>Substitutions</u> are container morphisms
- The <u>empty context</u> is $\mathbf{1} \triangleleft \mathbf{0}$

The container model (outlined in [Altenkirch and Kaposi, 2021])

- Contexts are set-containers $S_\Gamma : \mathrm{Set} \triangleleft P_\Gamma : S_\Gamma \to \mathrm{Set}$ with extension functor

$$[\![S_\Gamma \triangleleft P_\Gamma]\!] : \mathbf{Set} \to \mathbf{Set}$$

- Substitutions are container morphisms
- The empty context is $\mathbf{1} \triangleleft \mathbf{0}$
- Types in context $\Gamma = S_\Gamma \triangleleft P_\Gamma$ are generalised containers $S_A : \mathrm{Set} \triangleleft P_A : S_A \to |\int [\![\Gamma]\!]|$, with extension functor

$$[\![S_A \triangleleft P_A]\!] : (\textstyle\int [\![\Gamma]\!]) \to \mathbf{Set}.$$

The container model (outlined in [Altenkirch and Kaposi, 2021])

- Contexts are set-containers $S_\Gamma : \text{Set} \triangleleft P_\Gamma : S_\Gamma \to \text{Set}$ with extension functor

$$[\![S_\Gamma \triangleleft P_\Gamma]\!] : \mathbf{Set} \to \mathbf{Set}$$

- Substitutions are container morphisms
- The empty context is $\mathbf{1} \triangleleft \mathbf{0}$
- Types in context $\Gamma = S_\Gamma \triangleleft P_\Gamma$ are generalised containers $S_A : \text{Set} \triangleleft P_A : S_A \to |\textstyle\int [\![\Gamma]\!]|$, with extension functor

$$[\![S_A \triangleleft P_A]\!] : (\textstyle\int [\![\Gamma]\!]) \to \mathbf{Set}.$$

- Terms of type A in context Γ are dependent natural transformations from $[\![\Gamma]\!]$ to $[\![A]\!]$:

$$\int_{X:\text{Set}} (\gamma : [\![\Gamma]\!]\, X) \to [\![A]\!](X, \gamma)$$

The container model (outlined in [Altenkirch and Kaposi, 2021])

- Contexts are set-containers $S_\Gamma : \mathrm{Set} \triangleleft P_\Gamma : S_\Gamma \to \mathrm{Set}$ with extension functor

$$[\![S_\Gamma \triangleleft P_\Gamma]\!] : \mathbf{Set} \to \mathbf{Set}$$

- Substitutions are container morphisms
- The empty context is $\mathbf{1} \triangleleft \mathbf{0}$
- Types in context $\Gamma = S_\Gamma \triangleleft P_\Gamma$ are generalised containers $S_A : \mathrm{Set} \triangleleft P_A : S_A \to |\int [\![\Gamma]\!]|$, with extension functor

$$[\![S_A \triangleleft P_A]\!] : (\textstyle\int [\![\Gamma]\!]) \to \mathbf{Set}.$$

- Terms of type A in context Γ are dependent natural transformations from $[\![\Gamma]\!]$ to $[\![A]\!]$:

$$\int_{X:\mathrm{Set}} (\gamma : [\![\Gamma]\!]\, X) \to [\![A]\!](X, \gamma)$$

- Context extension is given by $\Gamma.A = S_A \triangleleft P_A^X$

Presheaf model vs Container model

Presheaf model

- Contexts: $\mathbf{C}^{\mathrm{op}} \to \mathbf{Set}$
- Substitutions: natural transformations
- Types: $(\int \Gamma)^{\mathrm{op}} \to \mathbf{Set}$
- Terms:

$\int_{X:\mathrm{Set}} (\gamma : \Gamma\, X) \to A(X, \gamma)$

- Context extension: $\Gamma.A\, X$

$= \sum(\rho : \Gamma\, X)(A\,(X, \rho))$

Container model

- Contexts: $\mathbf{Set} \to \mathbf{Set}$
- Substitutions: container morphisms
- Types: $(\int [\![\Gamma]\!]) \to \mathbf{Set}$
- Terms:

$\int_{X:\mathrm{Set}} (\gamma : [\![\Gamma]\!]\, X) \to [\![A]\!](X, \gamma)$

- Context extension: $[\![\Gamma.A]\!]\, X$

$= \sum(\rho : [\![\Gamma]\!]\, X)([\![A]\!]\,(X, \rho))$

Coherence issues in the absence of UIP

1. $Ty\colon \mathbf{Con}^{\mathrm{op}} \to \cancel{\mathbf{Set}}\ \mathbf{Gpd}$

   The **collection of types is a groupoid**, not an h-set. Additional coherence laws need to be checked.

$$
\begin{tikzcd}
 & & A[\sigma][\delta][\gamma] \arrow[dll] \arrow[drr] & & \\
A[\sigma][\delta \circ \gamma] \arrow[dr] & & & & A[\sigma \circ \delta][\gamma] \arrow[dl] \\
 & A[\sigma \circ (\delta \circ \gamma)] \arrow[rr] & & A[(\sigma \circ \delta) \circ \gamma] &
\end{tikzcd}
$$

$$
\begin{tikzcd}
 & A[id_\Gamma][\gamma] \arrow[dl] \arrow[dr] & \\
A[id_\Gamma \circ \gamma] \arrow[rr] & & A[\gamma]
\end{tikzcd}
$$

$$
\begin{tikzcd}
 & A[\gamma][id_\Gamma] \arrow[dl] \arrow[dr] & \\
A[\gamma \circ id_\Gamma] \arrow[rr] & & A[\gamma]
\end{tikzcd}
$$

Coherence issues in the absence of UIP

1. $Ty \colon \mathbf{Con}^{\mathrm{op}} \to \cancel{\mathbf{Set}}\ \mathbf{Gpd}$
   The **collection of types is a groupoid**, not an h-set. Additional coherence laws need to be checked.

$$\begin{tikzcd}
 & A[\sigma][\delta][\gamma] \arrow[dl] \arrow[dr] & \\
A[\sigma][\delta \circ \gamma] \arrow[d] & & A[\sigma \circ \delta][\gamma] \arrow[d] \\
A[\sigma \circ (\delta \circ \gamma)] \arrow[rr] & & A[(\sigma \circ \delta) \circ \gamma]
\end{tikzcd}$$

$$\begin{tikzcd}
 & A[id_\Gamma][\gamma] \arrow[dl] \arrow[dr] & \\
A[id_\Gamma \circ \gamma] \arrow[rr] & & A[\gamma]
\end{tikzcd}
\qquad
\begin{tikzcd}
 & A[\gamma][id_\Gamma] \arrow[dl] \arrow[dr] & \\
A[\gamma \circ id_\Gamma] \arrow[rr] & & A[\gamma]
\end{tikzcd}$$

2. **Functor laws do not hold strictly**, but only up to isomorphism, due to definition of type substitution $A[\gamma]$:

$$\begin{tikzcd}
S_{A[\gamma]} \arrow[r, "snd"] \arrow[d, "fst"'] \arrow[dr, phantom, "\lrcorner", very near start] & S_A \arrow[d, "P^s_A"] \\
S_\Delta \arrow[r, "\gamma_s"'] & S_\Gamma
\end{tikzcd}
\quad \lhd \quad
\begin{tikzcd}
P_\Gamma\,(\gamma_s(fst\ s)) \arrow[r, "P^f_A(snd\ s)"] \arrow[d, "\gamma_p(fst\ s)"'] & P^X_A(snd\ s) \arrow[d, "\mathrm{inr}"] \\
P_\Delta(fst\ s) \arrow[r, "\mathrm{inl}"'] \arrow[ur, phantom, "\ulcorner", very near end] & P^X_{A[\gamma]}\ s
\end{tikzcd}$$

How do we solve the issues?

$$
\begin{tikzcd}
 & {} \arrow[dl] \arrow[dr] & \\
\text{View model as set-CwF} \arrow[d] & & \text{View model as `higher' CwF [Kraus, 2021, Uemura, 2023]} \arrow[d] \\
\text{Strictification} & & \text{Prove extra coherence laws} \arrow[d] \\
 & & \text{Interpretation of syntax is unclear!}
\end{tikzcd}
$$

How do we solve the issues?

View model as set-CwF

View model as ‘higher’ CwF
[Kraus, 2021, Uemura, 2023]

Strictification

we do both

Prove extra coherence laws

Interpretation of syntax is unclear!

Strictified container model

- We use an inductive-recursive universe U : Set, $El : U \rightarrow$ Set.

Strictified container model

- We use an inductive-recursive universe U : Set, $El : U \to$ Set.
- Contexts are codes for set-containers

$$S^U_\Gamma : U \triangleleft P^U_\Gamma : El\, S^U_\Gamma \to U$$

Strictified container model

- We use an inductive-recursive universe U : Set, $El\colon U \to$ Set.
- Contexts are codes for set-containers

$$S_\Gamma^U\colon U \lhd P_\Gamma^U\colon El\, S_\Gamma^U \to U$$

- Types in context Γ are codes for generalised containers, together with a substitution—we delay substitution

$$(\Gamma \xrightarrow{\delta} \Delta, S_B^U\colon U \lhd P_B^U\colon El\, S_B^U \to |\textstyle\int [\![\Delta]\!]|^U)$$

$B[\delta]$

Strictified container model

- We use an inductive-recursive universe U : Set, El : $U \to$ Set.
- Contexts are codes for set-containers

$$S^U_\Gamma : U \lhd P^U_\Gamma : El\, S^U_\Gamma \to U$$

- Types in context Γ are codes for generalised containers, together with a substitution—we delay substitution

$$(\Gamma \xrightarrow{\delta} \Delta, S^U_B : U \lhd P^U_B : El\, S^U_B \to |\textstyle\int [\![\Delta]\!]|^U)$$

$B[\delta]$

- Type substitution can now be defined as

$$(B[\delta])[\gamma] :\equiv B[\delta \circ \gamma]$$

Strictified container model

- We use an inductive-recursive universe U : Set, $El\colon U \to$ Set.
- <u>Contexts</u> are codes for set-containers

$$S^U_\Gamma \colon U \lhd P^U_\Gamma \colon El\, S^U_\Gamma \to U$$

- <u>Types</u> in context Γ are codes for generalised containers, together with a substitution—we delay substitution

$$(\Gamma \xrightarrow{\delta} \Delta, S^U_B \colon U \lhd P^U_B \colon El\, S^U_B \to |\textstyle\int[\![\Delta]\!]|^U)$$

$B[\delta]$

- Type substitution can now be defined as

$$(B[\delta])[\gamma] \coloneqq B[\delta \circ \gamma]$$

Now, the collection of types is an h-set, and functor laws hold strictly.

Contributions & ongoing work

- Worked out details of 'higher' container model outlined in [Altenkirch and Kaposi, 2021], including extra coherence laws
- Finalizing details of strictified container model
- Constructing Π-types, Σ-types, universe in both versions
- Started a formalisation of the 'higher' container model in Cubical Agda

Contributions & ongoing work

- Worked out details of 'higher' container model outlined in [Altenkirch and Kaposi, 2021], including extra coherence laws
- Finalizing details of strictified container model
- Constructing Π-types, Σ-types, universe in both versions
- Started a formalisation of the 'higher' container model in Cubical Agda

Thank you!

References

Altenkirch, T., Capriotti, P., Dijkstra, G., Kraus, N., and Nordvall Forsberg, F. (2018).
Quotient inductive-inductive types.
In Baier, C. and Dal Lago, U., editors, *FoSSACS*, pages 293–310. Springer.

Altenkirch, T. and Kaposi, A. (2021).
A container model of type theory.
In *TYPES 2021*.

Kraus, N. (2021).
Internal ∞-categorical models of dependent type theory : Towards 2ltt eating hott.
Symposium on Logic in Computer Science (LICS 2021), pages 1–14.

Uemura, T. (2023).
A general framework for the semantics of type theory.
Mathematical Structures in Computer Science, 33(3):134–179.