Data Types with Symmetries via Action Containers

HoTT/UF 2025

Philipp Joram Niccolò Veltri

Tallinn University of Technology, Estonia

2025-04-16

Overview

Goal of the talk

Introduce *action containers* to model data types with symmetries

Contents

- Good ol' Containers
  - Endofunctors and algebraic data types
  - Containers for polynomial functors
- Action containers
  - Construction via universal property
  - Closure properties
- 2-categorical interpretation:
  - Equality of container morphisms is structured
  - Interpretation as 2-endofunctors of groupoids

Containers: presentation of polynomials

Model of polymorphic data types

type constructors $:=$ endofunctors $F, G : \mathbf{Set} \to \mathbf{Set}$

polymorphic functions $:=$ natural transformations $\alpha : F \Rightarrow G$

Containers: presentation of polynomials

Model of polymorphic data types

type constructors := endofunctors $F, G : \mathbf{Set} \to \mathbf{Set}$

polymorphic functions := natural transformations $\alpha : F \Rightarrow G$

The nice class of *polynomial* endofunctors is described by *containers*:

a container

$$(S \triangleleft P)$$

$$S : \mathbf{Set}, P : S \to \mathbf{Set}$$

its interpretation as a polynomial

$$[\![S \triangleleft P]\!](X) := \sum_{s:S} (P(s) \to X)$$

Non-polynomial endofunctors

Caveat

Not all interesting functors are covered by this framework.

Non-polynomial endofunctors

Caveat

Not all interesting functors are covered by this framework.

Example

Cyclic lists are not polynomial:

$$\mathrm{Cyc}(X) := \sum_{n:\mathbb{N}} X^n/{\sim} \qquad \text{where} \qquad (x_1, \ldots, x_n) \sim (x_n, x_1, \ldots, x_{n-1})$$

Same for unordered pairs, finite multisets, ...

Action containers

Definition

An action container $F = (S \triangleleft P \triangleright^{\sigma} G)$ consists of

- shapes: a set S
- positions: a family of sets $P : S \to \mathbf{Set}$
- symmetries: a family of groups $G : S \to \mathbf{Group}$
- actions: a family of group actions: for each $s : S$, σ_s is an action of G_s on P_s

Action containers

Definition

An action container $F = (S \triangleleft P \triangleright^{\sigma} G)$ consists of

shapes	a set S
positions	a family of sets $P : S \to \mathbf{Set}$
symmetries	a family of groups $G : S \to \mathbf{Group}$
actions	a family of group actions: for each $s : S$, σ_s is an action of G_s on P_s

Intuition

Symmetries tell us under which permutations the contained data is invariant.

Action containers

Definition

An action container $F = (S \lhd P \rhd^{\sigma} G)$ consists of

- shapes: a set S
- positions: a family of sets $P : S \to \mathbf{Set}$
- symmetries: a family of groups $G : S \to \mathbf{Group}$
- actions: a family of group actions: for each $s : S$, σ_s is an action of G_s on P_s

Intuition

Symmetries tell us under which permutations the contained data is invariant.

Interpretation

$$[\![S \lhd P \rhd^{\sigma} G]\!](X) := \sum_{s:S} (P_s \to X) / \sim_s \qquad v \sim_s w := \exists g : G_s .\, v = w \circ \sigma_s(g)$$

Example

Cyclic lists come from a $\mathbb{Z}$-action on finite sets:

$$\mathsf{Cyc} = (n : \mathbb{N} \triangleleft \mathsf{Fin}(n) \triangleright^{\sigma_n} \mathbb{Z}) \qquad \sigma_n : \mathbb{Z} \to \mathfrak{S}(\mathsf{Fin}(n))$$

where σ_n is generated from the successor automorphism,

$$\begin{aligned} \mathsf{suc}_n &: \mathsf{Fin}(n) \simeq \mathsf{Fin}(n) \\ \mathsf{suc}_n(x) &:= x + 1 \mod n \end{aligned} \qquad \sigma_n(k) := \underbrace{\mathsf{suc}_n \circ \cdots \circ \mathsf{suc}_n}_{k \text{ times}}$$

Example

Cyclic lists come from a $\mathbb{Z}$-action on finite sets:

$$\mathrm{Cyc} = (n : \mathbb{N} \triangleleft \mathrm{Fin}(n) \triangleright^{\sigma_n} \mathbb{Z}) \qquad \sigma_n : \mathbb{Z} \to \mathfrak{S}(\mathrm{Fin}(n))$$

where σ_n is generated from the successor automorphism,

$$\begin{aligned} \mathrm{suc}_n &: \mathrm{Fin}(n) \simeq \mathrm{Fin}(n) \\ \mathrm{suc}_n(x) &:= x + 1 \mod n \end{aligned} \qquad \sigma_n(k) := \underbrace{\mathrm{suc}_n \circ \cdots \circ \mathrm{suc}_n}_{k \text{ times}}$$

Similar approach for finite multisets, unordered tuples, etc.

Categories of containers

Recall

The category of *Good ol' Containers*[1] is that of *families of sets*:

$$\mathcal{G} \simeq \mathsf{Fam}(\mathbf{Set}^{\mathbf{op}}) \simeq \int_{S:\mathbf{Set}} \textstyle\prod_S \mathbf{Set}^{\mathbf{op}}$$

[1] Abbott, Altenkirch, and Ghani, "Containers: Constructing strictly positive types".

Categories of containers

Recall

The category of *Good ol' Containers*[1] is that of *families of sets*:

$$\mathcal{G} \simeq \mathsf{Fam}(\mathbf{Set}^{\mathbf{op}}) \simeq \int_{S:\mathbf{Set}} \prod_S \mathbf{Set}^{\mathbf{op}}$$

Using machinery of *displayed categories* makes this very modular:

- Get the right notion of morphism for free
- Aligns with the primitives of type theory, makes formalization feasible

[1] Abbott, Altenkirch, and Ghani, "Containers: Constructing strictly positive types".

Category of action containers

Definition

The category of action containers is

$$\mathbf{ActionCont} := \mathrm{Fam}(\mathbf{Action})$$

where **Action** is the total category of *group actions*

$$\mathbf{Action} := \int_{G:\mathbf{Group}} \int_{P:\mathbf{Set}^{\mathbf{op}}} \mathrm{GroupHom}(G, \mathfrak{S}(P))$$

Corollary

Fam(**Action**) *is the free coproduct completion of* **Action**. *It is thus closed under (arbitrary) coproducts and products, and exponentiation by constants.*

A model of strictly positive types

Action containers model non-inductive single-variable strictly positive types.[2]

[2] Abbott, Altenkirch, and Ghani, "Containers: Constructing strictly positive types".

A model of strictly positive types

Action containers model non-inductive single-variable strictly positive types.[2]

- strictly positive: closure under products $F \times G$, coproducts $F + G$ and constant exponentiation F^J.
- single-variable: extension to *parametrized* containers is straightforward
- non-inductive: we are working on finding smallest μF and largest νF fixpoint

[2] Abbott, Altenkirch, and Ghani, "Containers: Constructing strictly positive types".

Properties of the interpretation

Action containers are inspired by quotient containers:[3]

Quotient containers are the subtype of action containers with *faithful* actions.

[3] Abbott, Altenkirch, Ghani, and McBride, "Constructing Polymorphic Programs with Quotient Types".

[4] That's why morphisms of quotient container are equivalence classes!

Properties of the interpretation

Action containers are inspired by quotient containers:[3]

> Quotient containers are the subtype of action containers with *faithful* actions.

Caveat

Interpretation $[\![-]\!] : \mathbf{ActionCont} \to \mathrm{Endo}(\mathbf{Set})$ is not fully faithful.

Reason

Quotients in **Set** identify too many morphisms.[4]

[3] Abbott, Altenkirch, Ghani, and McBride, "Constructing Polymorphic Programs with Quotient Types".

[4] That's why morphisms of quotient container are equivalence classes!

Properties of the interpretation

Action containers are inspired by quotient containers:[3]

> Quotient containers are the subtype of action containers with *faithful* actions.

Caveat

Interpretation $[\![-]\!] : \mathbf{ActionCont} \to \mathrm{Endo}(\mathbf{Set})$ is not fully faithful.

Reason

Quotients in **Set** identify too many morphisms.[4]

Fix

- Do not quotient, but relate morphisms by 2-cells
- Interpret action containers in *2-endofunctors of groupoids*.
- Go via *symmetric containers*

---

[3] Abbott, Altenkirch, Ghani, and McBride, "Constructing Polymorphic Programs with Quotient Types".

[4] That's why morphisms of quotient container are equivalence classes!

Symmetric containers

Definition (Gylterud, "Symmetric Containers")

A *symmetric container* $(S \triangleleft P)$ consists of

shapes an *h-groupoid* S

positions a *function* $P : S \to \mathrm{hSet}$

with interpretation in pseudofunctors of h-groupoids:

$$[\![S \triangleleft P]\!](X) :\equiv \sum_{s:S} P(s) \to X$$

Intuition

Symmetries are paths between shapes.

2-categories of containers

Symmetric containers naturally form a 2-category:

2-cells := the h-set of *homotopies* of container morphisms

Interpretation is a 2-functor:

$$[\![-]\!] : \mathbf{SymmCont} \to \mathsf{PsFun}(\mathsf{hGpd}, \mathsf{hGpd})$$

[5] Hofstra and Karvonen, "Inner automorphisms as 2-cells".

2-categories of containers

Symmetric containers naturally form a 2-category:

$$\text{2-cells} := \text{the h-set of } \textit{homotopies} \text{ of container morphisms}$$

Interpretation is a 2-functor:

$$[\![-]\!] : \textbf{SymmCont} \to \mathrm{PsFun}(\mathrm{hGpd}, \mathrm{hGpd})$$

Action containers form a 2-category as well:

- 2-cells arise naturally from "group homomorphisms up to conjugation"[5]
- correspond closely to a similar relation for quotient containers

[5] Hofstra and Karvonen, "Inner automorphisms as 2-cells".

Delooping of containers

Any group action determines a single-shape symmetric container:

- a group G defines a 1-object h-groupoid $\mathbb{B}G$ (a HIT)
- a G-action σ defines a family $\bar{\mathbb{B}}\sigma : \mathbb{B}G \to \mathsf{hSet}$

Delooping of containers

Any group action determines a single-shape symmetric container:

- a group G defines a 1-object h-groupoid $\mathbb{B}G$ (a HIT)
- a G-action σ defines a family $\bar{\mathbb{B}}\sigma : \mathbb{B}G \to \mathrm{hSet}$

Theorem

The above extends to a locally fully-faithful 2-functor

$$\mathbb{B}^* : \textbf{ActionCont} \to \textbf{SymmCont}$$
$$\mathbb{B}^*(S \triangleleft P \triangleright^{\sigma} G) = \left(\sum_{s:S} \mathbb{B}G_s \triangleleft \bar{\mathbb{B}}\sigma_s \right)$$

- classifies morphisms of action containers
- lets us construct symmetric containers in practice

Example

The cyclic list container is isomorphic to a bundle over the circle,

$$\mathsf{Cyc} \cong \big((n, x : \mathbb{N} \times S^1) \lhd \mathsf{Cover}_n(x)\big)$$

where $\mathsf{Cover}_n : S^1 \to \mathsf{hSet}$ is the n-fold cover of S^1.

(Co)inductive types

Want to present (co)inductive data types as fixpoints of substitution:

$$\mu F \cong F[F[F[\ldots]]]$$

Substitution should correspond to composition of container functors

$$[\![\mathbb{B}^* F[G]]\!] \simeq [\![\mathbb{B}^* F]\!] \circ [\![\mathbb{B}^* G]\!]$$

Candidates for $\mu F, \nu F$: **ActionCont** should follow from standard procedure[6]

[6] Abbott, Altenkirch, and Ghani, "Representing Nested Inductive Types Using W-Types".

(Co)inductive types

Want to present (co)inductive data types as fixpoints of substitution:

$$\mu F \cong F[F[F[\ldots]]]$$

Substitution should correspond to composition of container functors

$$[\![\mathbb{B}^* F[G]]\!] \simeq [\![\mathbb{B}^* F]\!] \circ [\![\mathbb{B}^* G]\!]$$

Candidates for $\mu F, \nu F$: **ActionCont** should follow from standard procedure[6]

Problem

Finding the correct definition of substitution is tricky, and action containers might be too strict to encode the necessary symmetries.

[6] Abbott, Altenkirch, and Ghani, "Representing Nested Inductive Types Using W-Types".

Strict symmetric containers

An h-groupoid G is strict if the set truncation map $|-|_0 : G \to \|G\|_0$ has a section

- "G's connected components are pointed"
- "G is a collection of groups"
- "G is skeletal"

[7] Mirrors the case for pointed, connected groupoids, aka groups.

Strict symmetric containers

An h-groupoid G is strict if the set truncation map $|-|_0 : G \to \|G\|_0$ has a section

- "G's connected components are pointed"
- "G is a collection of groups"
- "G is skeletal"

Fact

Maps preserving the strict structure form a h-set. Thus, strict groupoids form a 1-category.[7]

[7] Mirrors the case for pointed, connected groupoids, aka groups.

Avoiding 2-categories

Proposition

Strict symmetric containers (=shapes are strict groupoids) form a 1-category.

But groupoid of shapes in the image of $\mathbb{B}^*$ are strict, thus:

Theorem (WIP)

The 1-categories of action containers and strict symmetric containers are equivalent.

The plan

We can define substitution/μ-/ν-types/... for action containers if the corresponding constructions on symmetric containers lift to strict ones.

Conclusion

For a write-up, and a fair share of displayed 2-category theory in Cubical Agda:

`https://phijor.me/publications/2025-data-types-with-symmetries-via-action-containers.html`

Thank you!

Abbott, Michael, Thorsten Altenkirch, and Neil Ghani. “Containers: Constructing strictly positive types”. In: *Theoretical Computer Science* 342.1 (2005), pp. 3–27. ISSN: 0304-3975. DOI: 10.1016/j.tcs.2005.06.002.

— . “Representing Nested Inductive Types Using W-Types”. In: *Automata, Languages and Programming*. Springer Berlin Heidelberg, 2004, pp. 59–71. ISBN: 9783540278368. DOI: 10.1007/978-3-540-27836-8_8.

Abbott, Michael, Thorsten Altenkirch, Neil Ghani, and Conor McBride. “Constructing Polymorphic Programs with Quotient Types”. In: *Proc. of 7th Int. Conf. on Mathematics of Program Construction, MPC’04*. Ed. by Dexter Kozen and Carron Shankland. Vol. 3125. LNCS. Springer Berlin Heidelberg, 2004, pp. 2–15. ISBN: 9783540277644. DOI: 10.1007/978-3-540-27764-4_2.

Gylterud, Håkon Robbestad. “Symmetric Containers”. MA thesis. Department of Mathematics, Faculty of Mathematics and Natural Sciences, University of Oslo, 2011. URL: https://hdl.handle.net/10852/10740.

Hofstra, Pieter and Martti Karvonen. “Inner automorphisms as 2-cells”. In: *Theory and Applications of Categories* 42.2 (2024), pp. 19–40. eprint: http://www.tac.mta.ca/tac/volumes/42/2/42-02abs.html. URL: http://www.tac.mta.ca/tac/volumes/42/2/42-02.pdf.