Quotient inductive types as categorified containers

Thorsten Altenkirch

School of Computer Science, University of Nottingham, UK
psztxa@nottingham.ac.uk

Strictly positive types can be represented as containers, that is $\mathsf{S} : \mathsf{Set}$ and a family of positions $\mathsf{P} : \mathsf{S} \to \mathsf{Set}$ giving rise to a functor $\mathsf{S} \lhd \mathsf{P} : \mathsf{Set} \Rightarrow \mathsf{Set}$ given by $(\mathsf{S} \lhd \mathsf{P})\,\mathsf{X} = \Sigma\,\mathsf{s} : \mathsf{S} \bullet \mathsf{P}\,\mathsf{s} \to \mathsf{X}$, see [1]. [1]

What happens when we have quotient-inductive types (QITs) [2, 3, 5] ? A well known example are multisets, which can be defined as a QIT:

```
data MS (A : Set) where
  []  : MS A
  _::_ : A → MS A → MS A
  swap : a :: b :: xs ≡ b :: a :: xs
```

We can interpret multisets as truncated higher containers, that is we define $\mathsf{S} : \mathsf{Groupoid}$ using the groupoid whose objects are natural numbers and morphisms are permutations on the corresponding finite sets (ie. elements of the symmetry group).$\mathsf{P} : \mathsf{S} \to \mathsf{Set}$ maps its each number to a finite set of given size and the permutation to permutations on finite sets. However, to interpret this as a functor on sets we need to consider the set-truncation:

$$\mathsf{S} \lhd \mathsf{P} : \mathsf{Set} \to \mathsf{Set}$$
$$(\mathsf{S} \lhd \mathsf{P})\,\mathsf{A} = \lVert\, \Sigma\,\mathsf{s} : \mathsf{S} \bullet \mathsf{P}\,\mathsf{s} \to \mathsf{A} \,\rVert_0$$

Indeed, this notion corresponds to what Gylterud called *symmetric container* in [4].

However, not all parametric QITs can be modelled this way, counterexamples are propositional truncation:

```
data ||_|| (A : Set) : Set where
  |_|       : A → ‖ A ‖
  is-prop : (a b : ‖ A ‖) → a ≡ b
```

and finite sets which are a simple extension of finite multisets by contraction:

```
data FS (A : Set) where
  []  : FS A
  _::_ : A → FS A → FS A
  swap : a :: b :: xs ≡ b :: a :: xs
  contr : a :: a :: xs ≡ a :: xs
```

However, it turns out that we can model both by what Gylterud calls *categorified container*. These are given by a category S and a covariant presheaf $\mathsf{P} : \mathsf{S} \Rightarrow \mathsf{Set}$. We now define [2] the semantics as a a colimit:

[1] We use fantasy agda where _⇒_ stands for functors.

[2] In Fantasy agda we identify the name of a category with the type of objects and use the same name for the homsets. Functors can be applied to objects and morphisms.

```
data (S ◁ P) (A : Set) : Set where
  _,_ : (s : S) → (P s → A) → (S ◁ P) A
  eq : (f : S s₀ s₁) → (s₀ , g ∘ P f) ≡ (s₁ , g)
```

Note that symmetric container arise as a special case if S turns out to be a groupoid. Both propositional truncation and finite sets are categorified container:

propositional truncation We define S as follows: there are two objects 1 ,2 : S and two morphisms 0 ,1 : S (1 ,2) but no equations. The naming suggests that P is mapping 1 ,2 to the corresponding sets and 0 ,1 to the corresponding elements of 2. Now we define [[_]] = S ◁ P and ‖_‖ : A → ‖ A ‖ as [a] = (1 , (λ tt → a)). We can derive is-prop : [a] ≡ [b] using eq with both 1 and 2.

finite sets Similar to the description of MS we define S as the category whose objects are natural numbers but the morphisms are given by inverses of surjections on the corresponding finite sets. We can now model contr using the surjection n + 2 → n + 1 which identifies the two new elements but is the identity everywhere else.

We conjecture that all parametric QITs can be represented using categoriified container and we are investigating how the container can be systematically derived from the schematic definition of the QIT. We also consider a variation of the W-types as the initial algebra of a categorified container which would be a good candidate for a QW-type. We also hope that we can model Quotient-Inductive-Inductive types (QIITs) in this setting.

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 *International Conference on Foundations of Software Science and Computation Structures*, pages 293–310. Springer International Publishing Cham, 2018.

[3] Marcelo P Fiore, Andrew M Pitts, and SC Steenkamp. Quotients, inductive types, and quotient inductive types. *Logical Methods in Computer Science*, 18, 2022.

[4] Håkon Robbestad Gylterud. Symmetric containers. Master's thesis, 2011.

[5] Ambrus Kaposi, András Kovács, and Thorsten Altenkirch. Constructing quotient inductive-inductive types. *Proceedings of the ACM on Programming Languages*, 3(POPL):1–24, 2019.