Formalising Inductive & Coinductive Containers

Stefania Damato, Thorsten Altenkirch, Axel Ljungström

ITP '25

1st October 2025

Once upon a time . . .

Available online at www.sciencedirect.com
SCIENCE @ DIRECT®
ELSEVIER
Theoretical Computer Science 342 (2005) 3–27
Theoretical Computer Science
www.elsevier.com/locate/tcs

Containers: Constructing strictly positive types

Michael Abbott[a], Thorsten Altenkirch[b,*], Neil Ghani[c]
[a] Diamond Light Source, Rutherford Appleton Laboratory, UK
[b] School of Computer Science and Information Technology, Nottingham University, UK
[c] Department of Mathematics and Computer Science, University of Leicester, UK

Abstract

We introduce the notion of a *Martin-Löf category*—a locally cartesian closed category with disjoint coproducts and initial algebras of container functors (the categorical analogue of W-types)—and then establish that nested strictly positive inductive and coinductive types, which we call *strictly positive types*, exist in any Martin-Löf category.

Central to our development are the notions of *containers* and *container functors*. These provide a new conceptual analysis of data structures and polymorphic functions by exploiting dependent type theory as a convenient way to define constructions in Martin-Löf categories. We also show that morphisms between containers can be full and faithfully interpreted as polymorphic functions (i.e. natural transformations) and that, in the presence of W-types, all strictly positive types (including nested inductive and coinductive types) give rise to containers.
© 2005 Elsevier B.V. All rights reserved.

Keywords: Type theory; Category theory; Container functors; W-Types; Induction; Coinduction; Initial algebras; Final coalgebras

2. Background

2.1. The categorical semantics of dependent types

This paper can be read in two ways (see Proposition 2.5):

(1) as a construction within the extensional type theory $\mathbf{MLW}^{\mathbf{ext}}$ (see [8]) with finite types, W-types, a proof of true $\neq$ false and no universes;

(2) as a construction in the internal language of locally cartesian closed categories with disjoint coproducts and initial algebras of container functors in one variable—we call these **Martin-Löf categories**.

Once upon a time . . .

Proposition 5.3. *Given a container* $F \equiv (S \rhd P, Q) \in \mathcal{G}_{I+1}$ *then*

$$[\![\mathsf{W}_S Q \rhd \mathrm{Pos}_{P,\mathsf{sup}^\mu}]\!] X \cong \mu Y.\ [\![F]\!](X, Y);$$

writing $\mu F \equiv (\mathsf{W}_S Q \rhd \mathrm{Pos}_{P,\mathsf{sup}^\mu})$ *we can conclude that* $[\![\mu F]\!] \cong \mu [\![F[-]]\!]$.

Easy enough

Proposition 5.4. *Given a container* $F \equiv (S \rhd P, Q) \in \mathcal{G}_{I+1}$ *then*

$$\left[\mathsf{M}_S Q \rhd \mathrm{Pos}_{P,\mathsf{sup}^\nu}\right] X \cong \nu Y.\ [\![F]\!](X, Y);$$

writing $\nu F \equiv (\mathsf{M}_S Q \rhd \mathrm{Pos}_{P,\mathsf{sup}^\nu})$ *we have* $[\![\nu F]\!] \cong \nu [\![F[-]]\!]$.

Confusing

I should formalise!

The punch line

- We formalised 'container functors preserve initial algebras & terminal coalgebras' in Cubical Agda.
- We improved the original result:

	original	new
type theory	extensional	intensional
homotopy level	h-set	any

decidable type-checking

containers in HoTT

Background: Containers (a.k.a. polynomial functors)

A **container** is given by a pair $S : \mathrm{Set}$, $P : S \to \mathrm{Set}$, written $S \triangleleft P$.

Background: Containers (a.k.a. polynomial functors)

A **container** is given by a pair $S : \mathrm{Set}$, $P : S \to \mathrm{Set}$, written $S \triangleleft P$.

Containers have a functorial interpretation. The **container functor** $[\![S \triangleleft P]\!] : \mathrm{Set} \to \mathrm{Set}$ is defined as:

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

Background : Containers (a.k.a. polynomial functors)

A **container** is given by a pair $S : \cancel{\mathrm{Set}}^{\,\mathrm{Type}}$, $P : S \to \cancel{\mathrm{Set}}^{\,\mathrm{Type}}$, written $S \triangleleft P$.

Containers have a functorial interpretation. The **container functor** $[\![S \triangleleft P]\!] : \cancel{\mathrm{Set}}^{\,\mathrm{Type}} \to \cancel{\mathrm{Set}}^{\,\mathrm{Type}}$ is defined as:

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

Type = wild cat. of types

Background: I-ary containers

An **I-ary container** is given by a pair $S : \mathsf{Type}$, $\underline{P} : I \to S \to \mathsf{Type}$, written $S \triangleleft \underline{P}$.

$$\overbrace{\mathsf{Type} \times \mathsf{Type} \times \cdots}^{I}$$

The **I-ary container functor** $[\![S \triangleleft \underline{P}]\!] : \mathsf{Type}^{I} \to \mathsf{Type}$ is defined as:

$$[\![S \triangleleft \underline{P}]\!] \, \underline{X} := \sum_{s : S} \left(\prod_{i : I} \underline{P} \, i \, s \to \underline{X} \, i \right)$$

Example : Lists

E.g. $F_{List} : \mathbf{Set}^2 \to \mathbf{Set}$

$$(A, X) \mapsto 1 + (A \times X)$$

∃ S, P, Q such that

$$\cong \sum_{s:S} (P s \longrightarrow A) \times (Q s \to X)$$

$$F_{List}(A, X) \cong [\![S \triangleleft (P, Q)]\!] (A, X)$$

And, $\mu X.\ F_{List}(A, X) \cong [\![\mathbb{N} \triangleleft \mathrm{Fin}]\!] A$ μ closure

$\nu X.\ F_{List}(A, X) \cong [\![\mathbb{N}_\infty \triangleleft \mathrm{Cofin}]\!] A$ ν closure

Coinductive types

Induction

$$
\begin{aligned}
&\mathsf{data}\ \mathbb{N} : \mathsf{Type}\ \mathsf{where} \\
&\quad \left.\begin{aligned} &\mathsf{zero} : \mathbb{N} \\ &\mathsf{succ} : \mathbb{N} \to \mathbb{N} \end{aligned}\right\} \text{constructors}
\end{aligned}
$$

$$
\begin{aligned}
&\mathsf{isEven} : \mathbb{N} \to \mathsf{Type} \\
&\mathsf{isEven}\ \mathsf{zero} = \top \\
&\mathsf{isEven}\ (\mathsf{suc}\ \mathsf{zero}) = \bot \\
&\mathsf{isEven}\ (\mathsf{suc}\ (\mathsf{suc}\ n)) = \mathsf{isEven}\ n
\end{aligned}
$$

pattern matching

Coinductive types

Induction

data $\mathbb{N}$: Type where
  zero : $\mathbb{N}$
  succ : $\mathbb{N} \to \mathbb{N}$

} constructors

isEven : $\mathbb{N} \to$ Type
isEven zero $= \top$
isEven (suc zero) $= \bot$
isEven (suc (suc n)) $=$ isEven n

pattern matching

Coinduction

record Stream (A : Type) : Type where
  coinductive
  field
    hd : A
    tl : Stream A

} destructors

from : $\mathbb{N} \to$ Stream $\mathbb{N}$
hd (from n) $= n$
tl (from n) $=$ from (suc n)

copattern matching

Coinduction in Agda

In vanilla Agda (without postulates):

✓ copattern matching

✓ guarded corecursion

✗ not enough extensionality e.g. no function extensionality

Coinduction in Agda

In vanilla Agda (without postulates):

✓ copattern matching

✓ guarded corecursion

✗ not enough extensionality e.g. no function extensionality

In Cubical Agda, funExt is provable.

This facilitates coinductive reasoning.

$$\mathsf{funExt} : ((x : A) \to f\ x \equiv g\ x) \to f \equiv g$$
$$\mathsf{funExt}\ p\ i\ x = p\ x\ i$$

Background: Agda & Cubical Agda

Agda is a dependently-typed proof assistant based on Martin-Löf type theory.
Propositional equality is an inductive family.

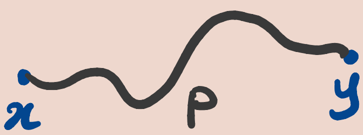
Background: Agda & Cubical Agda

Cubical Agda extends Agda with primitives from cubical type theory.

We have an interval pre-type I so that an equality $p : x \equiv_A y$ is now a function

$$p : I \to A$$

such that $p\ i0 = x$ and $p\ i1 = y$.



It has native support for the univalence axiom.

The statement (Prop. 5.4)

For $[\![S \triangleleft \underline{P}, Q]\!] : \mathrm{Type}^{I+1} \to \mathrm{Type}$, and for $\underline{X} : \mathrm{Type}^{I}$,

$$\left([\![M\ S\ Q \triangleleft \mathrm{PosM}]\!]\ \underline{X}\ ,\ \bullet \right)$$

is the terminal $[\![S \triangleleft \underline{P}, Q]\!] (\underline{X}, -)$-coalgebra.

The M-type

M is the type of non-wellfounded labelled trees.

```
record M (S : Type) (P : S → Type) : Type where
  coinductive
  field
    shape : S
    pos : P shape → M S P
```

finite & infinite paths

M is the universal type of strictly positive coinductive types.

Example : $\mathbb{N}\infty$

```
record ℕ∞ : Type where
  coinductive
  field
    pred∞ : Maybe ℕ∞
```

$0, 1, 2, \ldots : \mathbb{N}\infty$

$\infty : \mathbb{N}\infty$ and $\mathsf{pred}\infty\,(\infty) = \infty$

To represent $\mathbb{N}\infty$ via M, define :

$$
\begin{aligned}
S &= \top \uplus \top \\
Q\ (\mathsf{inl}\ \_) &= \bot \\
Q\ (\mathsf{inr}\ \_) &= \top,
\end{aligned}
$$

Then $M\,S\,Q \cong \mathbb{N}\infty$.

PosM : finite paths through an M-tree

```
data PosM : M S Q → Type where
  here : {m : M S Q} → PosM m
  below : {m : M S Q} {q : Q (shape m)} → PosM ((pos m) q) → PosM m
```

```
S = ⊤ ⊎ ⊤
Q (inl _) = ⊥
Q (inr _) = ⊤,
```

$M\,S\,Q \cong \mathbb{N}\infty$

tree representations	inl tt	inr tt → inl tt	inr tt (loop)
$\mathbb{N}\infty$	0	1 …	∞

PosM 0 = {here}

PosM 1 = {here, below (here)}

$|\text{PosM}\ \infty| = \infty$

Our Experience

- It was not obvious to us whether the original proof only worked for h-sets.

Our Experience

- It was not obvious to us whether the original proof only worked for h-sets.

- We had an issue with Agda's termination checker that meant we had to prove some things in a roundabout way.

Future work

- Main result of original paper talks about <u>containers</u> (not their functors) being closed under μ and ν. Requires more wild category theory.

- Containers in HoTT, for semantics of higher inductive types.

Future work

- Main result of original paper talks about <u>containers</u> (not their functors) being closed under μ and ν. Requires more wild category theory.

- Containers in HoTT, for semantics of higher inductive types.

THANK YOU!