Inductive Types for Free

Representing Nested Inductive Types using W-types

Michael Abbott (U. Leicester)

Thorsten Altenkirch (U. Nottingham)

Neil Ghani (U. Leicester)

Ideology

Ideology

- David Turner:
  Elementary Strong Functional Programming

Ideology

- David Turner:
  Elementary Strong Functional Programming
- Martin-Löf Type Theory

Ideology

- David Turner:
  Elementary Strong Functional Programming
- Martin-Löf Type Theory
- Types = sets, programs = total functions.

Ideology

- David Turner:
  Elementary Strong Functional Programming
- Martin-Löf Type Theory
- Types = sets, programs = total functions.
- Dependent types to avoid accidental partiality (e.g. hd).

Ideology

- David Turner:
  Elementary Strong Functional Programming
- Martin-Löf Type Theory
- Types = sets, programs = total functions.
- Dependent types to avoid accidental partiality (e.g. hd).
- E.g.: Conor McBride's Epigram system.

Plan of the talk

- Inductive and coinductive types.
- Container types for dummies.
- Properties of container types.
- W-types are sufficent for inductive types.
- Further work and applications
- Related work

Inductive and coinductive types

- Widespread in functional programming (e.g. Haskell)

$$\mathbf{data}\, \mathit{Lam} = \mathrm{Var}\, \mathit{Int} \mid \mathrm{App}\, \mathit{Lam}\, \mathit{Lam} \mid \mathrm{Lam}\, \mathit{Lam}$$

Inductive and coinductive types

- Widespread in functional programming (e.g. Haskell)

$$\mathbf{data}\,\textit{Lam} = \mathrm{Var}\,\textit{Int}\,|\,\mathrm{App}\,\textit{Lam}\,\textit{Lam}\,|\,\mathrm{Lam}\,\textit{Lam}$$

- Categorically: Initial algebra of a functor

$$\textit{Lam} = \mu X.1 + X \times X + X$$

μ vs. ν

- In a total setting $\mu \neq \nu$ (terminal coalgebras):

μ vs. ν

- In a total setting $\mu \neq \nu$ (terminal coalgebras):

  $\mu X.1 + A \times X$ finite lists

μ vs. ν

- In a total setting $\mu \neq \nu$ (terminal coalgebras):

$\mu X.1 + A \times X$	finite lists
$\nu X.1 + A \times X$	potentially infinite lists

μ vs. ν

- In a total setting $\mu \neq \nu$ (terminal coalgebras):

$\mu X.1 + A \times X$	finite lists
$\nu X.1 + A \times X$	potentially infinite lists
$\nu X.A \times X$	infinite lists

μ vs. ν

- In a total setting $\mu \neq \nu$ (terminal coalgebras):

$\mu X.1 + A \times X$	finite lists
$\nu X.1 + A \times X$	potentially infinite lists
$\nu X.A \times X$	infinite lists
$\mu X.A \times X$	

μ vs. ν

- In a total setting $\mu \neq \nu$ (terminal coalgebras):

$\mu X.1 + A \times X$	finite lists
$\nu X.1 + A \times X$	potentially infinite lists
$\nu X.A \times X$	infinite lists
$\mu X.A \times X$	$\emptyset$

μ vs. ν

- In a total setting $\mu \neq \nu$ (terminal coalgebras):

$\mu X.1 + A \times X$	finite lists
$\nu X.1 + A \times X$	potentially infinite lists
$\nu X.A \times X$	infinite lists
$\mu X.A \times X$	$\emptyset$

- μ, ν can be nested

μ vs. ν

- In a total setting $\mu \neq \nu$ (terminal coalgebras):

$\mu X.1 + A \times X$	finite lists
$\nu X.1 + A \times X$	potentially infinite lists
$\nu X.A \times X$	infinite lists
$\mu X.A \times X$	$\emptyset$

- μ, ν can be nested

$$\mathrm{FinTree} = \mu X.\mu Y.1 + X \times Y$$

μ vs. ν

- In a total setting $\mu \neq \nu$ (terminal coalgebras):

$\mu X.1 + A \times X$	finite lists
$\nu X.1 + A \times X$	potentially infinite lists
$\nu X.A \times X$	infinite lists
$\mu X.A \times X$	$\emptyset$

- μ, ν can be nested

$$\begin{aligned}\mathrm{FinTree} &= \mu X.\mu Y.1 + X \times Y \\ &= \mu X.\mathrm{List\,X}\end{aligned}$$

μ vs. ν

- In a total setting $\mu \neq \nu$ (terminal coalgebras):

$\mu X.1 + A \times X$	finite lists
$\nu X.1 + A \times X$	potentially infinite lists
$\nu X.A \times X$	infinite lists
$\mu X.A \times X$	$\emptyset$

- μ, ν can be nested

$$\begin{aligned} \mathrm{FinTree} &= \mu X.\mu Y.1 + X \times Y \\ &= \mu X.\mathrm{List}\,\mathrm{X} \\ \mathrm{DelayStream}\, A &= \nu X.\mu Y.Y + A \times X \end{aligned}$$

Axiomatically?

- How do we say?
  All inductive or coinductive datatypes

Axiomatically?

- How do we say?
  All inductive or coinductive datatypes
- Not every parametrized type has an initial algebra (or terminal coalgebra) in set theory, e.g.

Axiomatically?

- How do we say?
  All inductive or coinductive datatypes
- Not every parametrized type has an initial algebra (or terminal coalgebra) in set theory, e.g.

$$\mu X.X \to X \qquad ?$$

Axiomatically?

- How do we say?
  All inductive or coinductive datatypes
- Not every parametrized type has an initial algebra (or terminal coalgebra) in set theory, e.g.

$$\mu X.X \to X \qquad ?$$
$$\mu X.(X \to 2) \to 2 \quad ?$$

Axiomatically?

- How do we say?
  All inductive or coinductive datatypes
- Not every parametrized type has an initial algebra (or terminal coalgebra) in set theory, e.g.

$$\mu X.X \to X \qquad ?$$
$$\mu X.(X \to 2) \to 2 \quad ?$$

- Strictly positive types, e.g.

Axiomatically?

- How do we say?
  All inductive or coinductive datatypes
- Not every parametrized type has an initial algebra (or terminal coalgebra) in set theory, e.g.

$$\mu X.X \to X \qquad ?$$
$$\mu X.(X \to 2) \to 2 \quad ?$$

- Strictly positive types, e.g.
  $\mu X.1 + X + \mathrm{Nat} \to X$ ordinal notations

Axiomatically?

- How do we say?
  All inductive or coinductive datatypes
- Not every parametrized type has an initial algebra (or terminal coalgebra) in set theory, e.g.

$$\mu X.X \to X \qquad ?$$
$$\mu X.(X \to 2) \to 2 \quad ?$$

- Strictly positive types, e.g.
  $\mu X.1 + X + \mathrm{Nat} \to X$ ordinal notations
- How do we say?
  Strictly positive

Containers and W-types

- **Container types** are a syntax-free representation of strictly positive types.

Containers and W-types

- **Container types** are a syntax-free representation of strictly positive types.
- In an extensive LCCC: W-types ($\mathrm{W}\,S\,P = \mu X.\Sigma s \in S.P\,s \to X$) are sufficent to show that container types are closed under

Containers and W-types

- **Container types** are a syntax-free representation of strictly positive types.
- In an extensive LCCC: W-types ($\mathrm{W}\,S\,P = \mu X.\Sigma s \in S.P\,s \to X$) are sufficent to show that container types are closed under
  - μ (here)

Containers and W-types

- **Container types** are a syntax-free representation of strictly positive types.
- In an extensive LCCC: W-types ($\mathrm{W}\, S\, P = \mu X.\Sigma s \in S.P\, s \to X$) are sufficent to show that container types are closed under
  - μ (here)
  - ν (journal paper, submitted)

Containers and W-types

- **Container types** are a syntax-free representation of strictly positive types.
- In an extensive LCCC: W-types ($\mathrm{W}\, S\, P = \mu X.\Sigma s \in S.P\, s \to X$) are sufficent to show that container types are closed under
  - μ (here)
  - ν (journal paper, submitted)
- **Application**: Generic Programming e.g. see our work on derivatives of datatypes

Containers and W-types

- **Container types** are a syntax-free representation of strictly positive types.
- In an extensive LCCC: W-types ($\mathrm{W}\,S\,P = \mu X.\Sigma s \in S.P\,s \to X$) are sufficent to show that container types are closed under
  - μ (here)
  - ν (journal paper, submitted)
- **Application**: Generic Programming
  e.g. see our work on derivatives of datatypes
- **Application**: Small trusted cores
  e.g. for Epigram

Containers for dummies

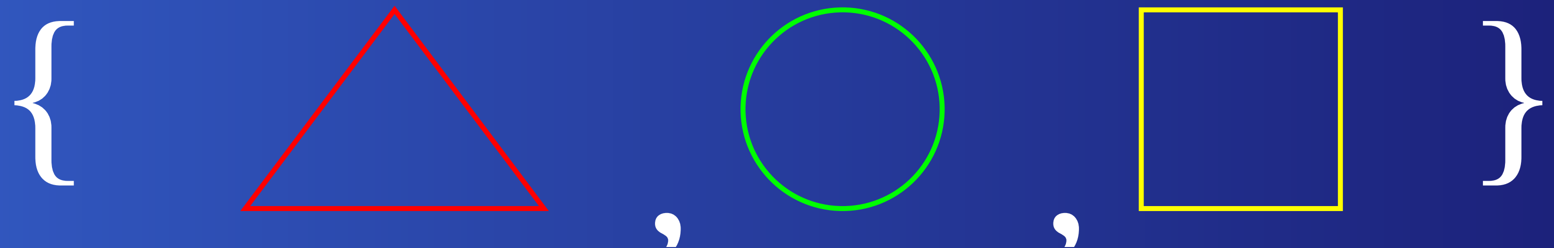
Containers for dummies

A container type is given by

Containers for dummies

A container type is given by

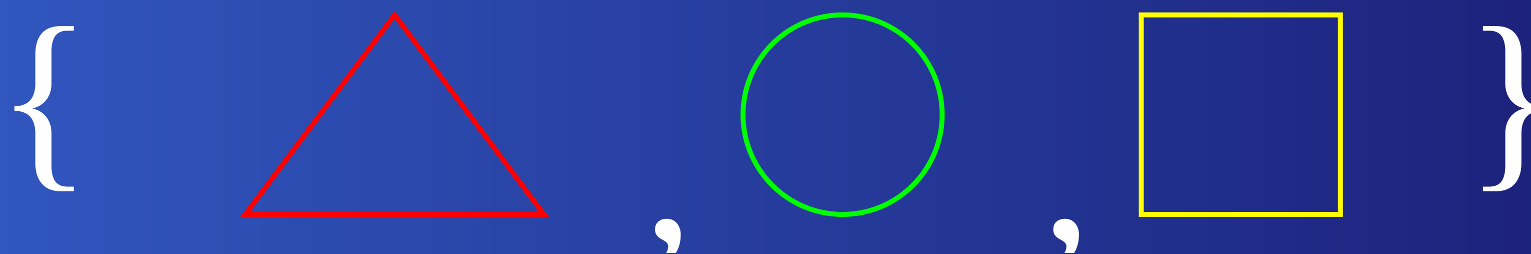
- A collection of shapes S, e.g.



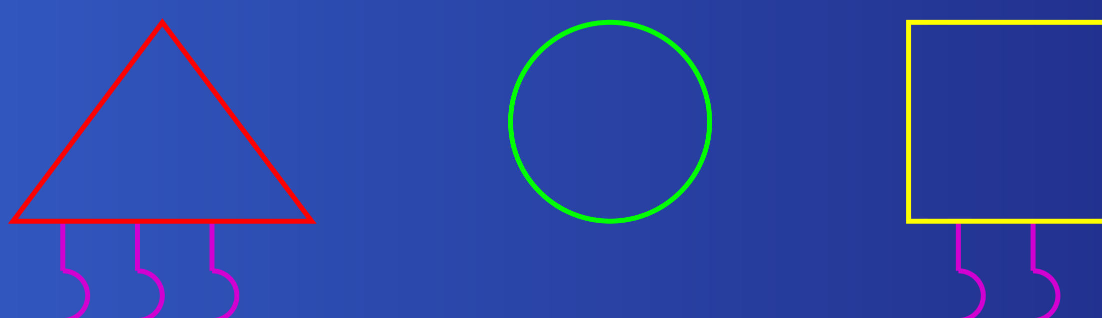
Containers for dummies

A container type is given by

- A collection of shapes S, e.g.



- An assignment of positions to shapes P, e.g.

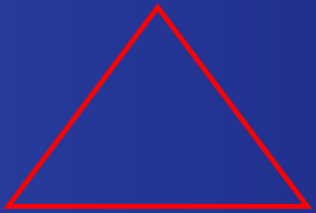


Containers for dummies . . .

We can use a container by

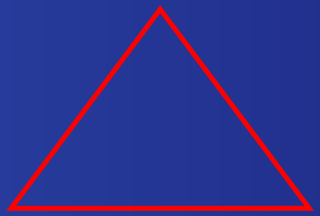
Containers for dummies . . .

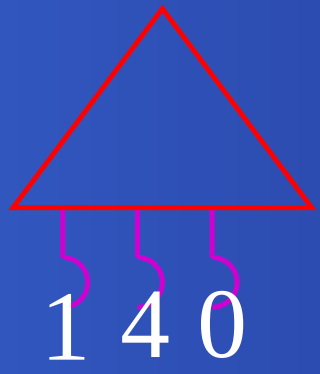
We can use a container by

- Choosing a shape, e.g. 

Containers for dummies . . .

We can use a container by

- Choosing a shape, e.g. 
- Filling the positions with payload (here natural numbers), e.g.



Container for scientists

Container for scientists

A container type $s \in S \triangleright P\, s$ is given by

- S a set of shapes
- P a family of sets of position, e.g. $P\, s$ is a set for any $s \in S$.

Container for scientists

A container type $s \in S \triangleright P\,s$ is given by

- S a set of shapes
- P a family of sets of position, e.g. $P\,s$ is a set for any $s \in S$.

The extension $[\![s \in S \triangleright P\,s]\!]$ of a container is the endofunctor

$$
\begin{aligned}
[\![s \in S \triangleright P\,s]\!] &: \mathbf{Set} \to \mathbf{Set} \\
[\![s \in S \triangleright P\,s]\!]\,X &= \Sigma s \in S.P\,s \to X
\end{aligned}
$$

Container for scientists

A container type $s \in S \triangleright P\,s$ is given by

- S a set of shapes
- P a family of sets of position, e.g. $P\,s$ is a set for any $s \in S$.

The extension $[\![s \in S \triangleright P\,s]\!]$ of a container is the endofunctor

$$\begin{aligned} [\![s \in S \triangleright P\,s]\!] &: \mathbf{Set} \to \mathbf{Set} \\ [\![s \in S \triangleright P\,s]\!]\,X &= \Sigma s \in S.P\,s \to X \end{aligned}$$

Straightforward extension to n-ary containers $s \in S \triangleright P_1\,s, P_2\,s, \ldots, P_n\,s$.

Example: Lists

$$\text{List}\, X \;=\; \mu Y.1 + X \times Y$$

Example: Lists

$$\begin{aligned} \mathrm{List}\,X &= \mu Y.1 + X \times Y \\ &\simeq \Sigma n \in \mathrm{Nat}.\{i \in \mathrm{Nat} \mid i < n\} \to X \end{aligned}$$

Example: Lists

$$\begin{aligned}\mathrm{List}\, X &= \mu Y.1 + X \times Y \\ &\simeq \Sigma n \in \mathrm{Nat}.\{i \in \mathrm{Nat} \mid i < n\} \to X\end{aligned}$$

$$S = \mathrm{Nat}$$

Example: Lists

$$
\begin{aligned}
\mathrm{List}\, X &= \mu Y.1 + X \times Y \\
&\simeq \Sigma n \in \mathrm{Nat}.\{i \in \mathrm{Nat} \mid i < n\} \to X
\end{aligned}
$$

$$
\begin{aligned}
S &= \mathrm{Nat} \\
P_n &= \{i \in \mathrm{Nat} \mid i < n\}
\end{aligned}
$$

Morphisms of containers

Given containers $S \triangleright P$ and $T \triangleright Q$ a morphism $f \triangleright u$ is given by

Morphisms of containers

Given containers $S \triangleright P$ and $T \triangleright Q$ a morphism $f \triangleright u$ is given by

$$\begin{aligned} f &\in S \to T \\ u &\in \Pi s \in S . Q\,(f\; s) \to P\; s \end{aligned}$$

Morphisms of containers

Given containers $S \rhd P$ and $T \rhd Q$ a morphism $f \rhd u$ is given by

$$\begin{aligned} f &\in S \to T \\ u &\in \Pi s \in S.Q\,(f\;s) \to P\;s \end{aligned}$$

its extension is the natural transformation

$$\begin{aligned} [\![f \rhd u]\!] &\in [\![S \rhd P]\!] \stackrel{\cdot}{\to} [\![T \rhd Q]\!] \\ [\![f \rhd u]\!](s,h) &= (f\;s, h \circ u\;s) \end{aligned}$$

Representation theorem

Theorem (AAG,FOSSACS 03)
The extension functor $[\![-]\!]$ is full and faithful.

Representation theorem

Theorem (AAG,FOSSACS 03)
The extension functor $[\![-]\!]$ is full and faithful.

Consequence: All polymorphic functions $\alpha_A : \mathrm{List}\, A \to \mathrm{List}\, A$ (i.e. natural transformations) are given by

Representation theorem

Theorem (AAG,FOSSACS 03)
The extension functor $[\![-]\!]$ is full and faithful.

Consequence: All polymorphic functions $\alpha_A : \mathrm{List}\, A \to \mathrm{List}\, A$ (i.e. natural transformations) are given by

- A length transformer $f \in \mathbb{N} \to \mathbb{N}$,

Representation theorem

Theorem (AAG,FOSSACS 03)
The extension functor $[\![-]\!]$ is full and faithful.

Consequence: All polymorphic functions $\alpha_A : \mathrm{List}\,A \to \mathrm{List}\,A$ (i.e. natural transformations) are given by

- A length transformer $f \in \mathbb{N} \to \mathbb{N}$,
- A where-did-you-come-from function $u \in \Pi n \in \mathbb{N}.P\,(f\,n) \to P\,n$.

Closure properties

Containers are closed under (*)

- Constant functors,
- Coproducts ($+$)
- Products ($\times$)
- Constant exponentation $F\,X = C \to G\,X$
- Composition of functors
- initial algebras (μ) [ICALP 04]
- terminal coalgebras (ν) [Journal paper]

Closure properties

Containers are closed under (*)

- Constant functors,
- Coproducts $(+)$
- Products $(\times)$
- Constant exponentation $F\,X = C \to G\,X$
- Composition of functors
- initial algebras (μ) [ICALP 04]
- terminal coalgebras (ν) [Journal paper]

(*) In any Martin-Löf category = LCCC (locally cartesian closed category) + W-types.

Coproducts of containers

Given containers

$$
\begin{aligned}
F\,X &= \Sigma s \in S.P\,s \to X \\
G\,X &= \Sigma t \in T.Q\,t \to X
\end{aligned}
$$

Coproducts of containers

Given containers

$$
\begin{aligned}
F\,X &= \Sigma s \in S.P\,s \to X \\
G\,X &= \Sigma t \in T.Q\,t \to X
\end{aligned}
$$

$$
F + G(X) = F\,X + G\,X
$$

Coproducts of containers

Given containers

$$\begin{aligned} F\,X &= \Sigma s \in S.P\,s \to X \\ G\,X &= \Sigma t \in T.Q\,t \to X \end{aligned}$$

$$\begin{aligned} F + G(X) &= F\,X + G\,X \\ &\simeq \Sigma u \in S + T.\left(\begin{cases} P(s) & \text{if } u = \mathrm{inl}(s) \\ Q(t) & \text{if } t = \mathrm{inr}(s) \end{cases}\right) \to X \end{aligned}$$

Products of containers

Given containers

$$
\begin{aligned}
F\,X &= \Sigma s \in S.P\,s \to X \\
G\,X &= \Sigma t \in T.Q\,t \to X
\end{aligned}
$$

Products of containers

Given containers

$$\begin{aligned} F\,X &= \Sigma s \in S.P\,s \to X \\ G\,X &= \Sigma t \in T.Q\,t \to X \end{aligned}$$

$$F \times G(X) = F\,X \times G\,X$$

Products of containers

Given containers

$$
\begin{aligned}
F\,X &= \Sigma s \in S. P\,s \to X \\
G\,X &= \Sigma t \in T. Q\,t \to X
\end{aligned}
$$

$$
\begin{aligned}
F \times G(X) &= F\,X \times G\,X \\
&\simeq \Sigma(s,t) \in S \times T. (P(s) + Q(t)) \to X
\end{aligned}
$$

Initial algebras (μ)

Given a 2-ary container

$$H\,X\,Y = \Sigma s \in S.P\,s \to X \times Q\,s \to Y$$

Initial algebras (μ)

Given a 2-ary container

$$H\,X\,Y = \Sigma s \in S.P\,s \to X \times Q\,s \to Y$$

$$\mu Y.H\,X\,Y \;=\; \Sigma t \in T.R\,t \to X$$

Initial algebras (μ)

Given a 2-ary container

$$H\,X\,Y = \Sigma s \in S.P\,s \to X \times Q\,s \to Y$$

$$\begin{aligned} \mu Y.H\,X\,Y &= \Sigma t \in T.R\,t \to X \\ T &= \mu Y.\Sigma s \in S.Q(s) \to Y \end{aligned}$$

Initial algebras (μ)

Given a 2-ary container

$$H\,X\,Y = \Sigma s \in S.P\,s \to X \times Q\,s \to Y$$

$$\begin{aligned} \mu Y.H\,X\,Y &= \Sigma t \in T.R\,t \to X \\ T &= \mu Y.\Sigma s \in S.Q(s) \to Y \\ &= W\,S\,Q \end{aligned}$$

Initial algebras (μ)

Given a 2-ary container

$$H\,X\,Y = \Sigma s \in S.P\,s \to X \times Q\,s \to Y$$

$$\begin{aligned}
\mu Y.H\,X\,Y &= \Sigma t \in T.R\,t \to X \\
T &= \mu Y.\Sigma s \in S.Q(s) \to Y \\
&= W\,S\,Q \\
R\,(s, f) &\simeq P\,s + \Sigma q \in Q\,s.R\,(f\,q)
\end{aligned}$$

Reply to referee comment

. . . Now, the above is a strictly positive defi nition so should have a least as well as a greatest solution which are not in general isomorphic.

Thus the corollary mentioned in the proof of 4.1 would be wrong and as a result the entire argument collapses.

I thus fear that the paper must be rejected; . . .

Reply to referee comment

. . . Now, the above is a strictly positive defi nition so should have a least as well as a greatest solution which are not in general isomorphic.

Thus the corollary mentioned in the proof of 4.1 would be wrong and as a result the entire argument collapses.

I thus fear that the paper must be rejected; . . .

Reply: *Since there are no infinite paths in a finite tree, there is only one solution to this isomorphism, the initial one.*

This is reflected in the proof of proposition 4.1!

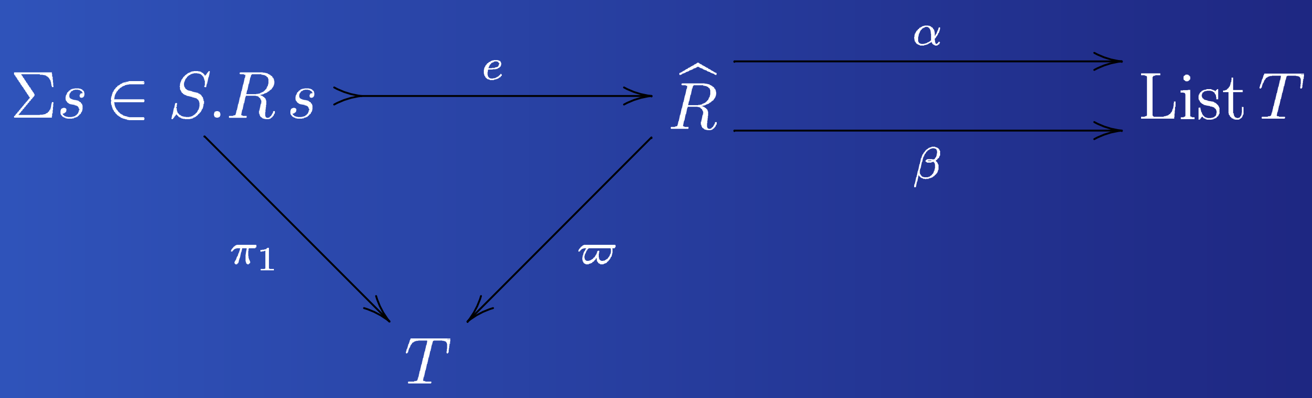
Constructing the iso

Constructing the iso

$$\begin{aligned}\hat{R} &= \mathrm{List}\,(\Sigma s \in S.(Q\,s \to T) \times Q\,s) \\ &\quad \times \Sigma s \in S.(Q\,s \to T) \times P\,s\end{aligned}$$

Constructing the iso

$$\begin{aligned}\hat{R} &= \text{List}\,(\Sigma s \in S.(Q\,s \to T) \times Q\,s)\\ &\quad \times \Sigma s \in S.(Q\,s \to T) \times P\,s\end{aligned}$$



Terminal coalgebras [Journal submiss

Terminal coalgebras [Journal submiss

- The same construction works for the ν-case.

Terminal coalgebras [Journal submiss

- The same construction works for the ν-case.
- Now we have to show that the solution is initial!

Terminal coalgebras [Journal submiss

- The same construction works for the ν-case.
- Now we have to show that the solution is initial!
- We need M-types, the dual of W-types.

Terminal coalgebras [Journal submiss

- The same construction works for the ν-case.
- Now we have to show that the solution is initial!
- We need M-types, the dual of W-types.
- However, M-types can be constructed from W-types.

Terminal coalgebras [Journal submiss

- The same construction works for the ν-case.
- Now we have to show that the solution is initial!
- We need M-types, the dual of W-types.
- However, M-types can be constructed from W-types.

See: *Containers - Constructing Strictly Positive Types* on my publication page.

Further work

- Quotient containers to model types like bags.
  First steps, see our MPC paper.
  Constructing Polymorphic Programs with Quotient Types
- Dependent containers
  Work in progress.

Related work

Joyal 86 *Foncteurs Analytiques et Espèces de Structures*

Jay 95 *A semantics for shape*

Dybjer 97 *Representing inductively defined sets by wellorderings in Martin-Löf's type theory*

Hoogendijk and de Moor 00 *Container Types Categorically*

Moerdijk and Palmgren 00 *Wellfounded Trees in Categories*

Hasegawa 02 *Two applications of analytic functors*

Gambino and Hyland 03 *Wellfounded Trees and Dependent Polynomial Functors*