A Container Model of Type Theory

Stefania Damato
j.w.w. Thorsten Altenkirch

University of Nottingham, UK

YaMCATS Meeting 32

14th September 2023

Motivation:
modelling inductive types

Functorial semantics, for ordinary inductive types

```
data ℕ : Set where
  zero : ℕ
  succ : ℕ → ℕ
```

Functorial semantics, for ordinary inductive types

```
data ℕ : Set where
  zero : ℕ
  succ : ℕ → ℕ
```

$\downarrow$

$\mathtt{zero} : \mathbb{N}^1$

$\mathtt{succ} : \mathbb{N}^{\mathbb{N}}$

Functorial semantics, for ordinary inductive types

```
data ℕ : Set where
  zero : ℕ
  succ : ℕ → ℕ
```

$\downarrow$

$\mathtt{zero} : \mathbb{N}^1$

$\mathtt{succ} : \mathbb{N}^{\mathbb{N}}$

$\downarrow$

$\mathtt{zero} \times \mathtt{succ} : \mathbb{N}^{1+\mathbb{N}}$

Functorial semantics, for ordinary inductive types

```
data ℕ : Set where
  zero : ℕ
  succ : ℕ → ℕ
```

$$\downarrow$$

$\mathtt{zero} : \mathbb{N}^1$

$\mathtt{succ} : \mathbb{N}^{\mathbb{N}}$

$$\downarrow$$

$$\mathtt{zero} \times \mathtt{succ} : \mathbb{N}^{1+\mathbb{N}}$$

$$\downarrow$$

$$\mathtt{zero} \times \mathtt{succ} : 1 + \mathbb{N} \to \mathbb{N}$$

Functorial semantics, for ordinary inductive types

```
data ℕ : Set where
  zero : ℕ
  succ : ℕ → ℕ
```

$\downarrow$

$\texttt{zero} : \mathbb{N}^1$

$\texttt{succ} : \mathbb{N}^{\mathbb{N}}$

$\downarrow$

$\texttt{zero} \times \texttt{succ} : \mathbb{N}^{1+\mathbb{N}}$

$\downarrow$

$\texttt{zero} \times \texttt{succ} : 1 + \mathbb{N} \to \mathbb{N}$

$\downarrow$

$$F_{\mathbb{N}} : \mathbf{Set} \to \mathbf{Set}$$
$$F_{\mathbb{N}}(X) :\equiv \mathbf{1} + X$$

Functorial semantics, for ordinary inductive types

```
data ℕ : Set where
  zero : ℕ
  succ : ℕ → ℕ
```

$\downarrow$

$\texttt{zero} : \mathbb{N}^1$

$\texttt{succ} : \mathbb{N}^{\mathbb{N}}$

$\downarrow$

$\texttt{zero} \times \texttt{succ} : \mathbb{N}^{1+\mathbb{N}}$

$\downarrow$

$\texttt{zero} \times \texttt{succ} : 1 + \mathbb{N} \to \mathbb{N}$

$\downarrow$

$F_{\mathbb{N}} : \textbf{Set} \to \textbf{Set}$

$F_{\mathbb{N}}(X) \coloneqq \mathbf{1} + X$

```
data C : Set where
    c : ((C → 2) → 2) → C
```

Functorial semantics, for ordinary inductive types

```
data ℕ : Set where
  zero : ℕ
  succ : ℕ → ℕ
```

$\downarrow$

$\mathtt{zero} : \mathbb{N}^1$

$\mathtt{succ} : \mathbb{N}^{\mathbb{N}}$

$\downarrow$

$\mathtt{zero} \times \mathtt{succ} : \mathbb{N}^{1+\mathbb{N}}$

$\downarrow$

$\mathtt{zero} \times \mathtt{succ} : 1 + \mathbb{N} \rightarrow \mathbb{N}$

$\downarrow$

$F_{\mathbb{N}} : \mathbf{Set} \rightarrow \mathbf{Set}$

$F_{\mathbb{N}}(X) := \mathbf{1} + X$

```
data C : Set where
    c : ((C → 2) → 2) → C
```

$\downarrow$

$\mathtt{c} : ((\mathtt{C} \rightarrow 2) \rightarrow 2) \rightarrow \mathtt{C}$

Functorial semantics, for ordinary inductive types

```
data ℕ : Set where
  zero : ℕ
  succ : ℕ → ℕ
```

$\downarrow$

$\mathtt{zero} : \mathbb{N}^1$

$\mathtt{succ} : \mathbb{N}^{\mathbb{N}}$

$\downarrow$

$\mathtt{zero} \times \mathtt{succ} : \mathbb{N}^{1+\mathbb{N}}$

$\downarrow$

$\mathtt{zero} \times \mathtt{succ} : 1 + \mathbb{N} \to \mathbb{N}$

$\downarrow$

$F_{\mathbb{N}} : \mathbf{Set} \to \mathbf{Set}$

$F_{\mathbb{N}}(X) := \mathbf{1} + X$

```
data C : Set where
    c : ((C → 2) → 2) → C
```

$\downarrow$

$\mathtt{c} : ((\mathtt{C} \to 2) \to 2) \to \mathtt{C}$

$\downarrow$

$F_C : \mathbf{Set} \to \mathbf{Set}$

$F_C(X) := (X \to \mathbf{2}) \to \mathbf{2}$

The W-type

```
data W (S : Set) (P : S → Set) : Set where
  sup : (s : S) → (P s → W S P) → W S P
```

The W-type

```
data W (S : Set) (P : S → Set) : Set where
  sup : (s : S) → (P s → W S P) → W S P
```

ℕ as a W-type

```
data ℕ : Set where
  zero : ℕ
  succ : ℕ → ℕ
```

The W-type

```
data W (S : Set) (P : S → Set) : Set where
  sup : (s : S) → (P s → W S P) → W S P
```

ℕ as a W-type

```
data ℕ : Set where
  zero : ℕ
  succ : ℕ → ℕ
```

$S := 1 + 1$

The W-type

```
data W (S : Set) (P : S → Set) : Set where
  sup : (s : S) → (P s → W S P) → W S P
```

ℕ as a W-type

```
data ℕ : Set where
  zero : ℕ
  succ : ℕ → ℕ
```

$$S := \mathbf{1} + \mathbf{1}$$

$$P(\mathsf{inl}\ \star) := \mathbf{0}$$

$$P(\mathsf{inr}\ \star) := \mathbf{1}$$

The W-type

```
data W (S : Set) (P : S → Set) : Set where
  sup : (s : S) → (P s → W S P) → W S P
```

ℕ as a W-type

```
data ℕ : Set where
  zero : ℕ
  succ : ℕ → ℕ
```

$\mathbb{N} \cong \mathtt{W}\,\mathtt{S}\,\mathtt{P}$.

$$\mathtt{S} := \mathbf{1} + \mathbf{1}$$

$$\mathtt{P}(\mathsf{inl}\,\star) := \mathbf{0}$$

$$\mathtt{P}(\mathsf{inr}\,\star) := \mathbf{1}$$

The W-type

```
data W (S : Set) (P : S → Set) : Set where
  sup : (s : S) → (P s → W S P) → W S P
```

ℕ as a W-type

```
data ℕ : Set where
  zero : ℕ
  succ : ℕ → ℕ
```

$$\mathsf{S} := \mathbf{1} + \mathbf{1}$$

$$\mathsf{P}(\mathrm{inl}\,\star) := \mathbf{0}$$

$$\mathsf{P}(\mathrm{inr}\,\star) := \mathbf{1}$$

$\mathbb{N} \cong \mathtt{W\,S\,P}$.

$$z : \mathtt{W\,S\,P}$$

$$z := \mathrm{sup}\,(\mathrm{inl}\,\star)\,(\lambda\,())$$

$$s : \mathtt{W\,S\,P} \longrightarrow \mathtt{W\,S\,P}$$

$$s\,n := \mathrm{sup}\,(\mathrm{inr}\,\star)\,(\lambda\,\_.n)$$

The W-type

```
data W (S : Set) (P : S → Set) : Set where
  sup : (s : S) → (P s → W S P) → W S P
```

ℕ as a W-type

```
data ℕ : Set where
  zero : ℕ
  succ : ℕ → ℕ
```

$\mathbb{N} \cong \mathtt{W\,S\,P}$.

$z : \mathtt{W\,S\,P}$

$z := \mathrm{sup}\,(\mathrm{inl}\,\star)\,(\lambda\,())$

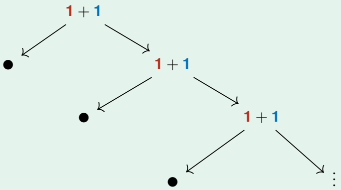
$s : \mathtt{W\,S\,P} \longrightarrow \mathtt{W\,S\,P}$

$s\,n := \mathrm{sup}\,(\mathrm{inr}\,\star)\,(\lambda\,\_\,.n)$

$\mathtt{S} := \mathbf{1} + \mathbf{1}$

$\mathtt{P}(\mathrm{inl}\,\star) := \mathbf{0}$

$\mathtt{P}(\mathrm{inr}\,\star) := \mathbf{1}$



Containers (a.k.a. polynomial functors)

Definition

A *container* is a pair S : Set, $P\colon\ S \to$ Set, written as $S \lhd P$.

Containers (a.k.a. polynomial functors)

Definition

A *container* is a pair S : Set, $P\colon S \to$ Set, written as $S \triangleleft P$.

Definition

Extension functor $[\![S \triangleleft P]\!]\colon \mathbf{Set} \to \mathbf{Set}$ is defined by

$$[\![S \triangleleft P]\!]\, X := \sum(s : S)(P\, s \to X).$$

Containers (a.k.a. polynomial functors)

Definition

A *container* is a pair S : Set, P: $S \to$ Set, written as $S \lhd P$.

Definition

Extension functor $[\![S \lhd P]\!] : \mathbf{Set} \to \mathbf{Set}$ is defined by

$$[\![S \lhd P]\!]\, X := \sum (s : S)(P\, s \to X).$$

$\mathbb{N}$'s container representation

$$\mathrm{S} := \mathbf{1} + \mathbf{1}$$

$$\mathrm{P}(\mathrm{inl}\,\star) := \mathbf{0}$$

$$\mathrm{P}(\mathrm{inr}\,\star) := \mathbf{1}$$

$$[\![S \lhd P]\!] : \mathbf{Set} \to \mathbf{Set}$$

$$\begin{aligned} [\![S \lhd P]\!]\, X &= \sum (s : \mathbf{1} + \mathbf{1})((\lambda\, (\mathrm{inl}\,\star).\mathbf{0};\, (\mathrm{inr}\,\star).\mathbf{1}) \to X) \\ &\cong \mathbf{1} + X \end{aligned}$$

Containers (a.k.a. polynomial functors)

Definition

A *container* is a pair S : Set, $P\colon S \to$ Set, written as $S \triangleleft P$.

Definition

Extension functor $[\![S \triangleleft P]\!]\colon \mathbf{Set} \to \mathbf{Set}$ is defined by

$$[\![S \triangleleft P]\!]\, X := \sum(s : S)(P\, s \to X).$$

Containers enforce strict positivity semantically.

An overview of inductive types

Class of types	**Functor type**	**Category theory semantics**	**Type theoretic normal form**	**Universal type**
ordinary inductive types e.g. $\mathbb{N}$: Set	**Set** $\rightarrow$ **Set**	initial algebras of endofunctors on **Set**	containers	W-type

An overview of inductive types

Class of types	Functor type	Category theory semantics	Type theoretic normal form	Universal type
ordinary inductive types e.g. $\mathbb{N}$: Set	**Set** $\to$ **Set**	initial algebras of endofunctors on **Set**	containers	W-type
inductive families e.g. $\mathtt{Fin} : \mathbb{N} \to \mathrm{Set}$	$(\mathbf{I} \to \mathbf{Set}) \to (\mathbf{I} \to \mathbf{Set})$	initial algebras of endofuntors on $\mathbf{Set}^I$	indexed containers	WI-type

An overview of inductive types

Class of types	**Functor type**	**Category theory semantics**	**Type theoretic normal form**	**Universal type**
ordinary inductive types e.g. $\mathbb{N}$: Set	$\mathbf{Set} \to \mathbf{Set}$	initial algebras of endofunctors on **Set**	containers	W-type
inductive families e.g. $\mathtt{Fin} : \mathbb{N} \to \mathrm{Set}$	$(\mathbf{I} \to \mathbf{Set}) \to (\mathbf{I} \to \mathbf{Set})$	initial algebras of endofuntors on $\mathbf{Set}^I$	indexed containers	WI-type
QIITs e.g. $\mathtt{Con} : \mathrm{Set}$, $\mathtt{Ty} : \mathtt{Con} \to \mathrm{Set}$	?	?	?	?

Quotient inductive-inductive types

$$\text{QIITs} = \begin{cases} \textbf{Quotient} \text{ inductive types} \\ \textbf{Inductive-inductive} \text{ types} \end{cases}$$

Quotient inductive-inductive types

QIITs = $\begin{cases} \textbf{Quotient} \text{ inductive types} \\ \textbf{Inductive-inductive} \text{ types} \end{cases}$

Example 1.1: (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
```

Functorial semantics, for QIITs [Altenkirch et al., 2018]

Example 1.1

```
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
```

1. Category $\mathbf{A_0}$ of sorts.

Functorial semantics, for QIITs [Altenkirch et al., 2018]

Example 1.1

```
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
```

1. Category $\mathbf{A_0}$ of sorts.
2. Constructor specification. The n^{th} constructor is specified by two functors

$$L_n : \mathbf{A_n} \to \mathbf{Set},$$

$$R_n : \int L_n \to \mathbf{Set}.$$

Functorial semantics, for QIITs [Altenkirch et al., 2018]

Example 1.1

```
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
```

1. Category $\mathbf{A_0}$ of sorts.
2. Constructor specification. The n^{th} constructor is specified by two functors

$$L_n \colon \mathbf{A_n} \to \mathbf{Set},$$
$$R_n \colon \int L_n \to \mathbf{Set}.$$

3. Category of algebras. $\mathbf{A_{n+1}}$ is the category having objects of type $\sum(A : |\mathbf{A_n}|)(c \colon (x : L_n\, A) \to R_n(A, x))$.

Containerification

Goal: restrict

$$L_n \colon \mathbf{A_n} \to \mathbf{Set},$$
$$R_n \colon \int L_n \to \mathbf{Set}$$

to be container functors (+ other restrictions).

Containerification

Goal: restrict

$$L_n\colon \mathbf{A_n} \to \mathbf{Set},$$
$$R_n\colon \int L_n \to \mathbf{Set}$$

to be container functors (+ other restrictions).

Definition

Given category $\mathbf{C}$, a *generalised container* is a pair S : Set, $P\colon S \to |\mathbf{C}|$.

The *extension functor* $[\![S \triangleleft P]\!]\colon \mathbf{C} \to \mathbf{Set}$ is defined by

$$[\![S \triangleleft P]\!]\, X := \sum(s : S)(\mathbf{C}(P\, s, X)).$$

An overview of inductive types, revisited

Class of types	**Representation**	**Category theory semantics**	**Type theoretic normal form**	**Universal type**
ordinary inductive types e.g. $\mathbb{N}$: Set	functor $\mathbf{Set} \to \mathbf{Set}$	initial algebras of endofunctors on **Set**	containers	W-type
inductive families e.g. $\mathtt{Fin} : \mathbb{N} \to \mathrm{Set}$	functor $(\mathbf{I} \to \mathbf{Set}) \to (\mathbf{I} \to \mathbf{Set})$	initial algebras of endofuntors on $\mathbf{Set}^I$	indexed containers	WI-type
QIITs e.g. $\mathtt{Con} : \mathrm{Set}$, $\mathtt{Ty} : \mathtt{Con} \to \mathrm{Set}$	sequence of functors L_n and R_n and sequence of categories of dialgebras	initial object in last constructed category of dialgebras $\mathbf{A_n}$	representations constructed via generalised containers	? (QW-type)

The container model

Categories with families

Definition

A category with families (CwF) consists of:

- A category **C**, of contexts and context substitutions, having a terminal object.

Categories with families

Definition

A category with families (CwF) consists of:

- A category $\mathbf{C}$, of contexts and context substitutions, having a terminal object.
- A functor $Ty\colon \mathbf{C}^{\mathrm{op}} \to \mathbf{Set}$.

Categories with families

Definition

A category with families (CwF) consists of:

- A category $\mathbf{C}$, of contexts and context substitutions, having a terminal object.
- A functor $Ty\colon \mathbf{C}^{\mathrm{op}} \to \mathbf{Set}$.
- A functor $Tm\colon (\int Ty)^{\mathrm{op}} \to \mathbf{Set}$.

Categories with families

Definition

A category with families (CwF) consists of:

- A category $\mathbf{C}$, of contexts and context substitutions, having a terminal object.
- A functor $Ty\colon \mathbf{C}^{\mathrm{op}} \to \mathbf{Set}$.
- A functor $Tm\colon (\int Ty)^{\mathrm{op}} \to \mathbf{Set}$.
- For every $\Gamma : |\mathbf{C}|$ and $A : Ty(\Gamma)$,
  - an object $\Gamma.A : |\mathbf{C}|$
  - a morphism $p\colon \Gamma.A \to \Gamma$ in $\mathbf{C}$
  - and a term $q : Tm(\Gamma.A, A[p])$,

  with a certain universal property.

($-[f]$ denotes the action of Ty and Tm on a morphism f.)

Presheaf model

- Category **Con** whose
  - objects (contexts) are presheaves $\mathbf{C}^{\mathrm{op}} \to \mathbf{Set}$

Presheaf model

- Category **Con** whose
  - objects (contexts) are presheaves $\mathbf{C}^{\mathrm{op}} \to \mathbf{Set}$
  - morphisms (substitutions) are natural transformations

Presheaf model

- Category **Con** whose
  - objects (contexts) are presheaves $\mathbf{C}^{\mathrm{op}} \to \mathbf{Set}$
  - morphisms (substitutions) are natural transformations
  - terminal object (empty context) is the constant $\mathbf{1}$ presheaf.

Presheaf model

- Category **Con** whose
  - objects (contexts) are presheaves $\mathbf{C}^{op} \to \mathbf{Set}$
  - morphisms (substitutions) are natural transformations
  - terminal object (empty context) is the constant $\mathbf{1}$ presheaf.
- Types in context Γ are presheaves over $\int \Gamma$.

$$Ty : \mathbf{Con}^{op} \to \mathbf{Set}$$
$$Ty\,\Gamma := (\textstyle\int \Gamma)^{op} \to \mathbf{Set}$$

Presheaf model

- Category **Con** whose
  - objects (contexts) are presheaves $\mathbf{C}^{\mathrm{op}} \to \mathbf{Set}$
  - morphisms (substitutions) are natural transformations
  - terminal object (empty context) is the constant $\mathbf{1}$ presheaf.
- Types in context Γ are presheaves over $\int \Gamma$.

$$Ty \colon \mathbf{Con}^{\mathrm{op}} \to \mathbf{Set}$$
$$Ty\,\Gamma :\equiv (\textstyle\int \Gamma)^{\mathrm{op}} \to \mathbf{Set}$$

- Terms in context Γ of type A are dependent natural transformations from Γ to A.

$$Tm \colon (\textstyle\int Ty)^{\mathrm{op}} \to \mathbf{Set}$$
$$Tm\,(\Gamma, A) :\equiv \int_{X:\mathrm{Set}} (\gamma : \Gamma\, X) \to A(X, \gamma)$$

Presheaf model

- Category **Con** whose
  - objects (contexts) are presheaves $\mathbf{C}^{\mathrm{op}} \to \mathbf{Set}$
  - morphisms (substitutions) are natural transformations
  - terminal object (empty context) is the constant **1** presheaf.
- Types in context Γ are presheaves over $\int \Gamma$.

$$Ty : \mathbf{Con}^{\mathrm{op}} \to \mathbf{Set}$$
$$Ty\,\Gamma := (\textstyle\int \Gamma)^{\mathrm{op}} \to \mathbf{Set}$$

- Terms in context Γ of type A are dependent natural transformations from Γ to A.

$$Tm : (\textstyle\int Ty)^{\mathrm{op}} \to \mathbf{Set}$$
$$Tm\,(\Gamma, A) := \int_{X:\mathrm{Set}} (\gamma : \Gamma\,X) \to A(X, \gamma)$$

- Context extension $(\Gamma.A)\,X = \sum(\gamma : \Gamma\,X)(A(X, \gamma))$.

Container model

- Category **Con** whose
  - objects (contexts) are set-containers $S : \text{Set}, P : S \to \text{Set}$ with extension functor $[\![S \triangleleft P]\!] : \mathbf{Set} \to \mathbf{Set}$

Container model

- Category **Con** whose
  - objects (contexts) are set-containers $S : \mathrm{Set}, P : S \to \mathrm{Set}$ with extension functor $[\![S \triangleleft P]\!] : \mathbf{Set} \to \mathbf{Set}$
  - morphisms (substitutions) are container morphisms

Container model

- Category **Con** whose
  - objects (contexts) are set-containers $S : \mathrm{Set}, P : S \to \mathrm{Set}$ with extension functor $[\![S \triangleleft P]\!] : \mathbf{Set} \to \mathbf{Set}$
  - morphisms (substitutions) are container morphisms
  - terminal object (empty context) is $\mathbf{1} \triangleleft \mathbf{0}$.

Container model

- Category **Con** whose
  - objects (contexts) are set-containers S : Set, $P : S \to$ Set with extension functor $[\![S \triangleleft P]\!] : \mathbf{Set} \to \mathbf{Set}$
  - morphisms (substitutions) are container morphisms
  - terminal object (empty context) is $\mathbf{1} \triangleleft \mathbf{0}$.
- Types in context Γ are generalised containers S : Set, $P : S \to |\int[\![\Gamma]\!]|$ over $\int[\![\Gamma]\!]$, with extension functor

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

Container model

- Category **Con** whose
  - objects (contexts) are set-containers S : Set, P : $S \to$ Set with extension functor $[\![S \triangleleft P]\!] \colon \mathbf{Set} \to \mathbf{Set}$
  - morphisms (substitutions) are container morphisms
  - terminal object (empty context) is $\mathbf{1} \triangleleft \mathbf{0}$.
- Types in context Γ are generalised containers S : Set, $P \colon S \to |\int [\![\Gamma]\!]|$ over $\int [\![\Gamma]\!]$, with extension functor

$$[\![S \triangleleft P]\!] \colon (\textstyle\int [\![\Gamma]\!]) \to \mathbf{Set}.$$

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

$$Tm \colon (\textstyle\int Ty)^{\mathrm{op}} \to \mathbf{Set}$$

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

Container model

- Category **Con** whose
  - objects (contexts) are set-containers S : Set, $P\colon S \to$ Set with extension functor $[\![S \triangleleft P]\!]\colon \mathbf{Set} \to \mathbf{Set}$
  - morphisms (substitutions) are container morphisms
  - terminal object (empty context) is $\mathbf{1} \triangleleft \mathbf{0}$.
- Types in context Γ are generalised containers S : Set, $P\colon S \to |\int[\![\Gamma]\!]|$ over $\int[\![\Gamma]\!]$, with extension functor

$$[\![S \triangleleft P]\!]\colon (\textstyle\int[\![\Gamma]\!]) \to \mathbf{Set}.$$

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

$$Tm\colon (\textstyle\int Ty)^{\mathrm{op}} \to \mathbf{Set}$$

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

- Context extension
  $(\Gamma.A)\,X = \sum (\Gamma.S)(A.SA) \triangleleft \lambda\,(s\Gamma, sA).\,(A.PA)\,s\Gamma\,sA$

Presheaf model & container model

Presheaf model

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

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

Container model

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

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

To-dos

- Deal with coherence issues.

To-dos

- Deal with coherence issues.
  - Category of set-containers has a **groupoid** (as opposed to an h-set) of objects.

    $\hookrightarrow$ Add coherences to the CwF.

    $\hookrightarrow$ Strictify objects via an inductive-recursive universe:

```
data U : Set where
   nat : U

El : U → Set
El nat = ℕ
```

To-dos

- Deal with coherence issues.
  - Category of set-containers has a **groupoid** (as opposed to an h-set) of objects.

    $\hookrightarrow$ Add coherences to the CwF.

    $\hookrightarrow$ Strictify objects via an inductive-recursive universe:

    ```
    data U : Set where
       nat : U

    El : U → Set
    El nat = ℕ
    ```

  - Strictify pullbacks and pushouts (e.g. when proving $A[f \circ g] \equiv A[f][g]$).

To-dos

- Deal with coherence issues.
  - Category of set-containers has a **groupoid** (as opposed to an h-set) of objects.

    ↪ Add coherences to the CwF.
    ↪ Strictify objects via an inductive-recursive universe:

    ```
    data U : Set where
       nat : U

    El : U → Set
    El nat = ℕ
    ```

  - Strictify pullbacks and pushouts (e.g. when proving $A[f \circ g] \equiv A[f][g]$).
- For QIIT semantics, we need **Con** to be the category of generalised containers (as opposed to set-containers).

Related work

- Thorsten Altenkirch and Ambrus Kaposi's TYPES abstract 'A container model of type theory'.

Related work

- Thorsten Altenkirch and Ambrus Kaposi's TYPES abstract 'A container model of type theory'.
- Tamara von Glehn's polynomial functor model using comprehension categories.

Related work

- Thorsten Altenkirch and Ambrus Kaposi's TYPES abstract 'A container model of type theory'.
- Tamara von Glehn's polynomial functor model using comprehension categories.
- Bob Atkey and András Kovács's implementation of the same model as a CwF.

Summary

- QIITs combine set-truncated **equalities** with **induction-induction**.
- We can represent **QIITs semantically** as initial objects in a category of algebras.
- Containerification of QIIT semantics requires as a prerequisite the ability to **express any statement in type theory as a container**. This can be achieved by a **container model** of type theory.
- The container model is a **restricted version of the presheaf model**.

Summary

- QIITs combine set-truncated **equalities** with **induction-induction**.
- We can represent **QIITs semantically** as initial objects in a category of algebras.
- Containerification of QIIT semantics requires as a prerequisite the ability to **express any statement in type theory as a container**. This can be achieved by a **container model** of type theory.
- The container model is a **restricted version of the presheaf model**.

Thank you!

References I

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*.

Atkey, R. (2020).
Interpreting dependent types with containers.
Talk at the MSP101 seminar, University of Strathclyde, slides at `https://bentnib.org/docs/tt-in-containers.pdf`, code at `https://gist.github.com/bobatkey/0d1f04057939905d35699f1b1c323736`.

References II

Kovács, A. (2020).
Construction of the containers / polynomials cwf in agda.
Code at `https://gist.github.com/AndrasKovacs/cf7cc88f667c8f0087a4981f6be4eef8`.

von Glehn, T. (2015).
Polynomials and models of type theory.
PhD thesis, University of Cambridge.