

# A Container Model of Type Theory

Stefania Damato  
j.w.w. Thorsten Altenkirch

University of Nottingham, UK

YaMCATS Meeting 32

14<sup>th</sup> September 2023

Motivation:  
modelling inductive types

# Functorial semantics, for ordinary inductive types

data  $\mathbb{N}$  : Set where

zero :  $\mathbb{N}$

succ :  $\mathbb{N} \rightarrow \mathbb{N}$

# Functorial semantics, for ordinary inductive types

data  $\mathbb{N}$  : Set where

zero :  $\mathbb{N}$

succ :  $\mathbb{N} \rightarrow \mathbb{N}$

↓

zero :  $\mathbb{N}^1$

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

# Functorial semantics, for ordinary inductive types

data  $\mathbb{N}$  : Set where

zero :  $\mathbb{N}$

succ :  $\mathbb{N} \rightarrow \mathbb{N}$

↓

zero :  $\mathbb{N}^1$

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

↓

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

# Functorial semantics, for ordinary inductive types

data  $\mathbb{N}$  : Set where

zero :  $\mathbb{N}$

succ :  $\mathbb{N} \rightarrow \mathbb{N}$

↓

zero :  $\mathbb{N}^1$

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

↓

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

↓

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

# Functorial semantics, for ordinary inductive types

data  $\mathbb{N}$  : Set where

zero :  $\mathbb{N}$

succ :  $\mathbb{N} \rightarrow \mathbb{N}$

↓

zero :  $\mathbb{N}^1$

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

↓

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

↓

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

↓

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

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

# Functorial semantics, for ordinary inductive types

data  $\mathbb{N}$  : Set where

zero :  $\mathbb{N}$

succ :  $\mathbb{N} \rightarrow \mathbb{N}$

↓

zero :  $\mathbb{N}^1$

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

↓

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

↓

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

↓

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

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

data C : Set where

c :  $((C \rightarrow 2) \rightarrow 2) \rightarrow C$



# Functorial semantics, for ordinary inductive types

data  $\mathbb{N}$  : Set where

zero :  $\mathbb{N}$

succ :  $\mathbb{N} \rightarrow \mathbb{N}$

↓

zero :  $\mathbb{N}^1$

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

↓

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

↓

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

↓

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

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

data C : Set where

c :  $((C \rightarrow 2) \rightarrow 2) \rightarrow C$

↓

c :  $((C \rightarrow 2) \rightarrow 2) \rightarrow C$

# Functorial semantics, for ordinary inductive types

data  $\mathbb{N}$  : Set where

zero :  $\mathbb{N}$

succ :  $\mathbb{N} \rightarrow \mathbb{N}$

↓

zero :  $\mathbb{N}^1$

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

↓

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

↓

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

↓

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

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

data C : Set where

c :  $((C \rightarrow 2) \rightarrow 2) \rightarrow C$

↓

c :  $((C \rightarrow 2) \rightarrow 2) \rightarrow C$

↓

**$F_C : \mathbf{Set} \rightarrow \mathbf{Set}$**

**$F_C(X) := (X \rightarrow 2) \rightarrow 2$**

# 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
```

## $\mathbb{N}$ 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
```

## $\mathbb{N}$ 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
```

## $\mathbb{N}$ as a W-type

```
data N : Set where
  zero : N
  succ : N → N
  S := 1 + 1
  P(inl ★) := 0
  P(inr ★) := 1
```

# The W-type

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

## $\mathbb{N}$ as a W-type

```
data N : Set where
  zero : N
  succ : N → N
  S := 1 + 1
  P(inl ★) := 0
  P(inr ★) := 1
```

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

# The W-type

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

## $\mathbb{N}$ as a W-type

```
data N : Set where
  zero : N
  succ : N → N
  S := 1 + 1
  P(inl ★) := 0
  P(inr ★) := 1
```

$\mathbb{N} \cong \text{WSP}$ .

$z : \text{WSP}$

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

$s : \text{WSP} \rightarrow \text{WSP}$

$sn := \text{sup}(\text{inr } \star)(\lambda \_ . n)$



# The W-type

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

## $\mathbb{N}$ as a W-type

```
data  $\mathbb{N}$  : Set where
```

```
  zero :  $\mathbb{N}$ 
```

```
  succ :  $\mathbb{N} \rightarrow \mathbb{N}$ 
```

```
S := 1 + 1
```

```
P(inl ★) := 0
```

```
P(inr ★) := 1
```

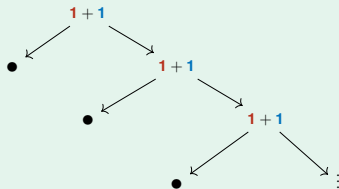
```
 $\mathbb{N} \cong \text{WSP}$ .
```

```
z : WSP
```

```
z := sup (inl ★) ( $\lambda ()$ )
```

```
s : WSP → WSP
```

```
sn := sup (inr ★) ( $\lambda _ . n$ )
```



# Containers (a.k.a. polynomial functors)

## Definition

A *container* is a pair  $S : \text{Set}, P : S \rightarrow \text{Set}$ , written as  $S \triangleleft P$ .

# Containers (a.k.a. polynomial functors)

## Definition

A *container* is a pair  $S : \mathbf{Set}, P : S \rightarrow \mathbf{Set}$ , written as  $S \triangleleft P$ .

## Definition

*Extension functor*  $\llbracket S \triangleleft P \rrbracket : \mathbf{Set} \rightarrow \mathbf{Set}$  is defined by

$$\llbracket S \triangleleft P \rrbracket X := \sum (s : S)(P s \rightarrow X).$$

# Containers (a.k.a. polynomial functors)

## Definition

A *container* is a pair  $S : \mathbf{Set}, P : S \rightarrow \mathbf{Set}$ , written as  $S \triangleleft P$ .

## Definition

*Extension functor*  $\llbracket S \triangleleft P \rrbracket : \mathbf{Set} \rightarrow \mathbf{Set}$  is defined by

$$\llbracket S \triangleleft P \rrbracket X := \sum (s : S)(P s \rightarrow X).$$

## $\mathbb{N}$ 's container representation

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

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

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

$$\llbracket S \triangleleft P \rrbracket : \mathbf{Set} \rightarrow \mathbf{Set}$$

$$\begin{aligned} \llbracket S \triangleleft P \rrbracket X &= \sum (s : \mathbf{1} + \mathbf{1})((\lambda (\text{inl } \star). \mathbf{0}; (\text{inr } \star). \mathbf{1}) \rightarrow X) \\ &\cong \mathbf{1} + X \end{aligned}$$

# Containers (a.k.a. polynomial functors)

## Definition

A *container* is a pair  $S : \mathbf{Set}, P : S \rightarrow \mathbf{Set}$ , written as  $S \triangleleft P$ .

## Definition

*Extension functor*  $\llbracket S \triangleleft P \rrbracket : \mathbf{Set} \rightarrow \mathbf{Set}$  is defined by

$$\llbracket S \triangleleft P \rrbracket X := \sum (s : S)(P s \rightarrow X).$$

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. $\mathbb{N} : \mathbf{Set}$	$\mathbf{Set} \rightarrow \mathbf{Set}$	initial algebras of endofunctors on $\mathbf{Set}$	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. $\mathbb{N} : \mathbf{Set}$	$\mathbf{Set} \rightarrow \mathbf{Set}$	initial algebras of endofunctors on $\mathbf{Set}$	containers	W-type
inductive families e.g. $\mathbf{Fin} : \mathbb{N} \rightarrow \mathbf{Set}$	$(\mathbf{I} \rightarrow \mathbf{Set}) \rightarrow (\mathbf{I} \rightarrow \mathbf{Set})$	initial algebras of endofunctors on $\mathbf{Set}^{\mathbf{I}}$	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. $\mathbb{N} : \mathbf{Set}$	$\mathbf{Set} \rightarrow \mathbf{Set}$	initial algebras of endofunctors on $\mathbf{Set}$	containers	W-type
inductive families e.g. $\mathbf{Fin} : \mathbb{N} \rightarrow \mathbf{Set}$	$(\mathbf{I} \rightarrow \mathbf{Set}) \rightarrow (\mathbf{I} \rightarrow \mathbf{Set})$	initial algebras of endofunctors on $\mathbf{Set}^{\mathbf{I}}$	indexed containers	WI-type
QITs e.g. $\mathbf{Con} : \mathbf{Set}$ , $\mathbf{Ty} : \mathbf{Con} \rightarrow \mathbf{Set}$	?	?	?	?



# Quotient inductive-inductive types

QIITs =  $\left\{ \begin{array}{l} \mathbf{Quotient} \text{ inductive types} \\ \mathbf{Inductive-inductive} \text{ types} \end{array} \right.$

# Quotient inductive-inductive types

QIITs =  $\left\{ \begin{array}{l} \text{Quotient inductive types} \\ \text{Inductive-inductive types} \end{array} \right.$

## 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
```

## 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.

## 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  $n^{\text{th}}$  constructor is specified by two functors

$$L_n : \mathbf{A}_n \rightarrow \mathbf{Set},$$

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

## 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  $n^{\text{th}}$  constructor is specified by two functors

$$L_n : \mathbf{A}_n \rightarrow \mathbf{Set},$$

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

- 3 Category of algebras.  $\mathbf{A}_{n+1}$  is the category having objects of type  $\sum (A : |\mathbf{A}_n|)(c : (x : L_n A) \rightarrow R_n(A, x))$ .

# Containerification

Goal: restrict

$$L_n: \mathbf{A}_n \rightarrow \mathbf{Set},$$
$$R_n: \int L_n \rightarrow \mathbf{Set}$$

to be container functors (+ other restrictions).

Goal: restrict

$$L_n: \mathbf{A}_n \rightarrow \mathbf{Set},$$
$$R_n: \int L_n \rightarrow \mathbf{Set}$$

to be container functors (+ other restrictions).

## Definition

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

The *extension functor*  $\llbracket S \triangleleft P \rrbracket : \mathbf{C} \rightarrow \mathbf{Set}$  is defined by

$$\llbracket S \triangleleft P \rrbracket X := \sum (s : S)(\mathbf{C}(P s, X)).$$

# An overview of inductive types, revisited

Class of types	Representation	Category theory semantics	Type theoretic normal form	Universal type
ordinary inductive types e.g. $\mathbb{N} : \mathbf{Set}$	functor $\mathbf{Set} \rightarrow \mathbf{Set}$	initial algebras of endofunctors on $\mathbf{Set}$	containers	W-type
inductive families e.g. $\mathbf{Fin} : \mathbb{N} \rightarrow \mathbf{Set}$	functor $(\mathbf{I} \rightarrow \mathbf{Set}) \rightarrow (\mathbf{I} \rightarrow \mathbf{Set})$	initial algebras of endofunctors on $\mathbf{Set}^{\mathbf{I}}$	indexed containers	WI-type
QITs e.g. $\mathbf{Con} : \mathbf{Set}$ , $\mathbf{Ty} : \mathbf{Con} \rightarrow \mathbf{Set}$	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

## Definition

A category with families (CwF) consists of:

- A category  $\mathbf{C}$ , of contexts and context substitutions, having a terminal object.

## Definition

A category with families (CwF) consists of:

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

## Definition

A category with families (CwF) consists of:

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

## Definition

A category with families (CwF) consists of:

- A category  $\mathbf{C}$ , of contexts and context substitutions, having a terminal object.
- A functor  $Ty: \mathbf{C}^{\text{op}} \rightarrow \mathbf{Set}$ .
- A functor  $Tm: (\int Ty)^{\text{op}} \rightarrow \mathbf{Set}$ .
- For every  $\Gamma : |\mathbf{C}|$  and  $A : Ty(\Gamma)$ ,
  - an object  $\Gamma.A : |\mathbf{C}|$
  - a morphism  $p: \Gamma.A \rightarrow \Gamma$  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  $\mathbf{C}^{\text{op}} \rightarrow \mathbf{Set}$

# Presheaf model

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

# Presheaf model

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



# Presheaf model

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

$$\text{Ty} : \mathbf{Con}^{\text{op}} \rightarrow \mathbf{Set}$$

$$\text{Ty } \Gamma := (f \Gamma)^{\text{op}} \rightarrow \mathbf{Set}$$

# Presheaf model

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

$$Ty: \mathbf{Con}^{\text{op}} \rightarrow \mathbf{Set}$$

$$Ty \Gamma := (\int \Gamma)^{\text{op}} \rightarrow \mathbf{Set}$$

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

$$Tm: (\int Ty)^{\text{op}} \rightarrow \mathbf{Set}$$

$$Tm(\Gamma, A) := \int_{X:\mathbf{Set}} (\gamma : \Gamma X) \rightarrow A(X, \gamma)$$

# Presheaf model

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

$$Ty: \mathbf{Con}^{\text{op}} \rightarrow \mathbf{Set}$$

$$Ty \Gamma := (\int \Gamma)^{\text{op}} \rightarrow \mathbf{Set}$$

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

$$Tm: (\int Ty)^{\text{op}} \rightarrow \mathbf{Set}$$

$$Tm(\Gamma, A) := \int_{X: \mathbf{Set}} (\gamma: \Gamma X) \rightarrow A(X, \gamma)$$

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

# Container model

- Category **Con** whose
  - objects (contexts) are set-containers  $S : \mathbf{Set}, P : S \rightarrow \mathbf{Set}$  with extension functor  $\llbracket S \triangleleft P \rrbracket : \mathbf{Set} \rightarrow \mathbf{Set}$

# Container model

- Category **Con** whose
  - objects (contexts) are set-containers  $S : \mathbf{Set}, P : S \rightarrow \mathbf{Set}$  with extension functor  $\llbracket S \triangleleft P \rrbracket : \mathbf{Set} \rightarrow \mathbf{Set}$
  - morphisms (substitutions) are container morphisms

# Container model

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

$$\llbracket S \triangleleft P \rrbracket : (f[\Gamma]) \rightarrow \mathbf{Set}.$$

# Container model

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

$$\llbracket S \triangleleft P \rrbracket : (f[\llbracket \Gamma \rrbracket]) \rightarrow \mathbf{Set}.$$

- Terms in context  $\Gamma$  of type  $A$  are dependent natural transformations from  $\llbracket \Gamma \rrbracket$  to  $\llbracket A \rrbracket$ .

$$Tm : (f Ty)^{\text{op}} \rightarrow \mathbf{Set}$$

$$Tm(\Gamma, A) := \int_{X:\mathbf{Set}} (\gamma : \llbracket \Gamma \rrbracket X) \rightarrow \llbracket A \rrbracket(X, \gamma)$$



# Container model

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

$$\llbracket S \triangleleft P \rrbracket : (\int \llbracket \Gamma \rrbracket) \rightarrow \mathbf{Set}.$$

- Terms in context  $\Gamma$  of type  $A$  are dependent natural transformations from  $\llbracket \Gamma \rrbracket$  to  $\llbracket A \rrbracket$ .

$$Tm : (\int Ty)^{\text{op}} \rightarrow \mathbf{Set}$$

$$Tm(\Gamma, A) := \int_{X:\text{Set}} (\gamma : \llbracket \Gamma \rrbracket X) \rightarrow \llbracket A \rrbracket(X, \gamma)$$

- Context extension

$$? (\Gamma.A) X = \sum (\Gamma.S)(A.SA) \triangleleft \lambda (s\Gamma, sA). (A.PA) s\Gamma sA \quad ?$$

## Presheaf model

- Contexts:  $\mathbf{C}^{\text{op}} \rightarrow \mathbf{Set}$
- Substitutions: natural transformations
- Types:  $(f \Gamma)^{\text{op}} \rightarrow \mathbf{Set}$
- Terms:  
 $\int_{X:\mathbf{Set}} (\gamma : \Gamma X) \rightarrow A(X, \gamma)$

## Container model

- Contexts:  $\mathbf{Set} \rightarrow \mathbf{Set}$
- Substitutions: container morphisms
- Types:  $(f \llbracket \Gamma \rrbracket) \rightarrow \mathbf{Set}$
- Terms:  
 $\int_{X:\mathbf{Set}} (\gamma : \llbracket \Gamma \rrbracket X) \rightarrow \llbracket A \rrbracket (X, \gamma)$

- Deal with coherence issues.

- 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 =  $\mathbb{N}$

- 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  $\rightarrow$  Set  
El nat =  $\mathbb{N}$
  - Strictify pullbacks and pushouts (e.g. when proving  $A[f \circ g] \equiv A[f][g]$ ).

- 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 QIT semantics, we need **Con** to be the category of generalised containers (as opposed to set-containers).

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

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



- 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**.

- 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](https://gist.github.com/bobatkey/0d1f04057939905d35699f1b1c323736).



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.