Specifying QIITs using Containers

Stefania Damato Thorsten Altenkirch

University of Nottingham, UK

Conference on
Homotopy Type Theory

24th May 2023

Motivation

Gain a better understanding of QIITs.

Motivation

Gain a better understanding of QIITs.

$$\text{QIITs} = \begin{cases} \textbf{Quotient} \text{ inductive types} & \text{allow set-truncated equality constructors} \\ & \\ & \end{cases}$$

Motivation

Gain a better understanding of QIITs.

$$\text{QIITs} = \begin{cases} \textbf{Quotient}\ \text{inductive types} & \text{allow set-truncated equality} \\ & \text{constructors} \\ \textbf{Inductive-inductive}\ \text{types} & A : \mathsf{Type} \\ & B : A \to \mathsf{Type} \end{cases}$$

Motivation

Example 1.0: Intrinsic syntax of type theory

```
data Con : Set
data Ty : Con → Set
data Sub : Con → Con → Set
data Tm : (Γ : Con) → Ty Γ → Set

data Con where
   ◇ : Con
   _,_: (Γ : Con) → Ty Γ → Con

data Ty where
   -[-] : Ty Δ → Sub Γ Δ → Ty Γ
   U : {Γ : Con} → Ty Γ
   El : {Γ : Con} → Tm Γ U → Ty Γ
   Π : {Γ : Con} (A : Ty Γ) →
        Ty (Γ , A) → Ty Γ
   [id] : A [id] ≡ A
   [][] : A[σ][ν] ≡ A[σ ∘ ν]
   ...
```

```
data Sub where
   id : {Γ : Con} → Sub Γ Γ
   _∘_ : {Γ Δ Θ : Con} → Sub Θ Δ → Sub Γ Θ →
         Sub Γ Δ
   ε : {Γ : Con} → Sub Γ ◇
   _,_ : {Γ Δ : Con} {A : Ty Δ} (σ : Sub Γ Δ)
         → Tm Γ A[σ] → Sub Γ (Δ , A)
   π₁ : {Γ Δ : Con} {A : Ty Δ} →
        Sub Γ (Δ , A) → Sub Γ Δ
   id∘ : id ∘ σ ≡ σ
   ∘id : σ ∘ id ≡ σ
   ...

data Tm where
   -[-] : {Γ Δ : Con} {A : Ty Δ} {σ : Sub Γ Δ}
          → Tm Δ A → (σ : Sub Γ Δ) → Tm Γ A[σ]
   π₂ : {Γ Δ : Con} {A : Ty Δ} →
        (σ : Sub Γ (Δ , A)) → Tm Γ A[π₁ σ]
   lam : {Γ : Con} {A : Ty Γ} {B : Ty (Γ , A)} →
         Tm (Γ , A) B → Tm Γ (Π A B)
   app : {Γ : Con} {A : Ty Γ} {B : Ty (Γ , A)} →
         Tm Γ (Π A B) → Tm (Γ , A) B
   πβ : app (lam t) ≡ t
   ...
```

Motivation

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
```

Specifications 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. $\texttt{Fin} : \mathbb{N} \to \mathrm{Set}$	$(\mathbf{I} \to \mathbf{Set}) \to (\mathbf{I} \to \mathbf{Set})$	initial algebras of endofuntors on $\mathbf{Set}^I$	indexed containers	WI-type

Specifications 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. $\texttt{Fin} : \mathbb{N} \to \mathrm{Set}$	$(\mathbf{I} \to \mathbf{Set}) \to (\mathbf{I} \to \mathbf{Set})$	initial algebras of endofuntors on $\mathbf{Set}^I$	indexed containers	WI-type
QIITs e.g. $\texttt{Con} : \mathrm{Set}$, $\texttt{Ty} : \texttt{Con} \to \mathrm{Set}$	?	?	?	?

Specifications 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. $\texttt{Fin} : \mathbb{N} \to \mathrm{Set}$	$(\mathbf{I} \to \mathbf{Set}) \to (\mathbf{I} \to \mathbf{Set})$	initial algebras of endofuntors on $\mathbf{Set}^I$	indexed containers	WI-type
QIITs e.g. $\texttt{Con} : \mathrm{Set}$, $\texttt{Ty} : \texttt{Con} \to \mathrm{Set}$	we have an alternative representation	?	?	?

QIIT semantics [Altenkirch et al., 2018] by example

Idea: Start with the sorts, and add constructors one by one.

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
```

QIIT semantics [Altenkirch et al., 2018] by example

Idea: Start with the sorts, and add constructors one by one.

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. $\mathbf{A_0} = \mathbf{Fam}\,(\mathrm{Set})$ has
    - objects of type $\sum(C : \mathrm{Set})(T : C \to \mathrm{Set})$,
    - morphisms $(C, T) \to (C', T')$ are functions $f : C \to C'$ and $g : (c : C) \to T\,c \to T'\,(f\,c)$.

QIIT semantics [Altenkirch et al., 2018] by example

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
```

2. Adding constructor $\diamond$: `Con`.

$$L_0 \colon \mathbf{A_0} \to \mathbf{Set}$$
$$L_0(C, T) :\equiv \mathbf{1}$$

$$R_0 \colon \int L_0 \to \mathbf{Set}$$
$$= \sum \sum (C : \mathrm{Set})(T \colon C \to \mathrm{Set})(L_0(C, T)) \to \mathbf{Set}$$
$$R_0(C, T, \star) :\equiv C$$

QIIT semantics [Altenkirch et al., 2018] by example

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
```

3. Category of algebras $\mathbf{A_1}$ has

   - objects of type $\sum\sum(C : \mathrm{Set})(T \colon C \to \mathrm{Set})(e \colon (x : L_0(C,T)) \to R_0(C,T,x))$ $\cong \sum\sum(C : \mathrm{Set})(T \colon C \to \mathrm{Set})(e : C)$
   - morphisms $(C, T, e) \to (C', T', e')$ are functions $f \colon C \to C'$ and $g \colon (c : C) \to T\ c \to T'\,(f\ c)$ such that $e' \equiv f(e)$.

QIIT semantics [Altenkirch et al., 2018] by example

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
```

4. Adding constructor `_,_ : (Γ : Con) (A : Ty Γ) → Con`.

$$L_1 : \mathbf{A_1} \to \mathbf{Set}$$
$$L_1(C, T, e) :\equiv \sum(\Gamma : C)(T\,\Gamma)$$

$$R_1 : \int L_1 \to \mathbf{Set}$$
$$= \sum\sum\sum(C : \mathrm{Set})(T : C \to \mathrm{Set})(e : C)(L_1(C, T, e)) \to \mathbf{Set}$$
$$R_1(C, T, e, \Gamma, A) :\equiv C$$

QIIT semantics [Altenkirch et al., 2018] by example

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
```

5. Category of algebras $\mathbf{A_2}$ has
   - objects of type $\sum\sum\sum(C : \mathsf{Set})(T \colon C \to \mathsf{Set})(e : C)(ex \colon (x : L_1(C, T, e)) \to R_1(C, T, e, x))$
     $\cong \sum\sum\sum(C : \mathsf{Set})(T \colon C \to \mathsf{Set})(e : C)(ex \colon \sum(\Gamma : C)(T\,\Gamma) \to C)$
   - morphisms $(C, T, e, ex) \to (C', T', e', ex')$ are functions $f \colon C \to C'$ and $g \colon (c : C) \to T\,c \to T'\,(f\,c)$ such that $e' \equiv f(e)$ and $ex' \circ (f\,\Gamma, g\,\Gamma\,t) \equiv f \circ ex(\Gamma, t)$,

and so on.

Specification of inductive types, revisited

Class of types	**Representation**	**Category theory semantics**	**Type theoretic normal form**	**Universal type**
ordinary inductive types e.g. $\mathbb{N} : \mathrm{Set}$	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 $\mathbf{A_n}$	?	?

Specification 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. $\texttt{Fin} : \mathbb{N} \to \mathrm{Set}$	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. $\texttt{Con} : \mathrm{Set}$, $\texttt{Ty} : \texttt{Con} \to \mathrm{Set}$	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)

Our contribution: restricting L_n and R_n

We require $L_n \colon \mathbf{A_n} \to \mathbf{Set}$ and $R_n \colon \int L_n \to \mathbf{Set}$ to be **generalised container functors** (+ other restrictions on R_n).

Our contribution: restricting L_n and R_n

We require $L_n \colon \mathbf{A_n} \to \mathbf{Set}$ and $R_n \colon \int L_n \to \mathbf{Set}$ to be **generalised container functors** (+ other restrictions on R_n).

Definition

A *generalised container* $S \triangleleft P$ over a category $\mathbf{C}$ is a pair $S : \mathrm{Set}$ and $P \colon S \to |\mathbf{C}|$.

Definition

The *generalised container extension functor* associated to $S \triangleleft P$ and having type $\mathbf{C} \to \mathbf{Set}$, is defined by

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

on objects $X : |\mathbf{C}|$.

Our contribution: restricting L_n and R_n

$$S_n : \mathrm{Set}$$
$$P_n : S_n \to |\mathbf{A}_n|$$

$$Q_n^X : S_n \to |\mathbf{A_n}|$$
$$Q_n^f : (s : S_n) \to \mathbf{A_n}(P_n\, s, Q_n^X\, s)$$

$$\begin{aligned} L_n &: \mathbf{A}_n \to \mathbf{Set} \\ L_n\, X &\cong [\![S_n \triangleleft P_n]\!]\, X \; = \; \textstyle\sum (s : S_n)(f : \mathbf{A_n}(P_n\, s, X)) \end{aligned}$$

$$\begin{aligned} R_n &: \textstyle\int L_n \to \mathbf{Set} \\ R_n\,(X, (s, f)) &\cong [\![S_n \triangleleft Q_n]\!]\,(X, (s, f)) \\ &\cong \textstyle\sum (h : \mathbf{A_n}(Q_n^X\, s, X))(h \circ Q_n^f\, s = f) \end{aligned}$$

Our contribution: restricting L_n and R_n

$$
\begin{aligned}
S_n &: \mathrm{Set} \\
P_n &: S_n \to |\mathbf{A}_n|
\end{aligned}
$$

$$
\begin{aligned}
Q_n^X &: S_n \to |\mathbf{A_n}| \\
Q_n^f &: (s : S_n) \to \mathbf{A_n}(P_n\, s, Q_n^X\, s)
\end{aligned}
$$

$$
\begin{aligned}
L_n &: \mathbf{A}_n \to \mathbf{Set} \\
L_n\, X &\cong [\![S_n \triangleleft P_n]\!]\, X = \sum(s : S_n)(f : \mathbf{A_n}(P_n\, s, X))
\end{aligned}
$$

$$
\begin{aligned}
R_n &: \textstyle\int L_n \to \mathbf{Set} \\
R_n\,(X, (s, f)) &\cong [\![S_n \triangleleft Q_n]\!]\,(X, (s, f)) \\
&\cong \sum(h : \mathbf{A_n}(Q_n^X\, s, X))(h \circ Q_n^f\, s = f)
\end{aligned}
$$

Discussion and future work

- QIITs combine set-truncated equalities with induction-induction.
- We can represent QIITs semantically as initial dialgebras.
- We propose a **specification of strictly positive QIITs**, which is more semantic than the one given by [Kaposi et al., 2019].
- What does a **universal QW-type** look like? One proposal given by [Fiore et al., 2021].

Discussion and future work

- QIITs combine set-truncated equalities with induction-induction.
- We can represent QIITs semantically as initial dialgebras.
- We propose a **specification of strictly positive QIITs**, which is more semantic than the one given by [Kaposi et al., 2019].
- What does a **universal QW-type** look like? One proposal given by [Fiore et al., 2021].

Thank you!

References

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*.

Fiore, M. P., Pitts, A. M., and Steenkamp, S. (2021).
Quotients, inductive types, and quotient inductive types.
Log. Methods Comput. Sci., 18.

Kaposi, A., Kovács, A., and Altenkirch, T. (2019).
Constructing quotient inductive-inductive types.
Proc. ACM Program. Lang., 3(POPL).