

Parametric Quantifiers for Dependent Type Theory

Andreas Nuyts^{*}, Andrea Vezzosi[‡], Dominique Devriese^{*}

^{*}KU Leuven, [‡]Chalmers University of Technology

TYPES 2017
ELTE, Budapest, Hungary
June 1, 2017

Parametricity

- Variable is **parametric** if **only used for type-checking**
⇒ free well-behavedness theorems.
- Well-studied in System F, System F ω , Haskell, ...

Parametricity in dependent types

- **Some** parametricity results carry over,
Takeuti (2001), Bernardy, Jansson and Paterson (2012), Krishnaswami and Dreyer (2013), Atkey, Ghani and Johann (2014)
- Some can be made provable **internally**,
Bernardy, Coquand and Moulin (2015), Guilhem Moulin's PhD (2016)
- Some are **lost**.

Results

- We formulate a **sound** dependent type system ParamDTT.
- We carry over “the” **remaining theorems** metatheoretically.
- We allow proving **additional theorems internally**.

Parametricity

- Variable is **parametric** if **only used for type-checking**
⇒ free well-behavedness theorems.
- Well-studied in System F, System F ω , Haskell, ...

Parametricity in dependent types

- **Some** parametricity results carry over,
Takeuti (2001), Bernardy, Jansson and Paterson (2012), Krishnaswami and Dreyer (2013), Atkey, Ghani and Johann (2014)
- Some can be made provable **internally**,
Bernardy, Coquand and Moulin (2015), Guilhem Moulin's PhD (2016)
- Some are **lost**.

Results

- We formulate a **sound** dependent type system ParamDTT.
- We carry over “the” **remaining theorems** metatheoretically.
- We allow proving **additional theorems internally**.

Parametricity

- Variable is **parametric** if **only used for type-checking**
⇒ free well-behavedness theorems.
- Well-studied in System F, System F ω , Haskell, ...

Parametricity in dependent types

- **Some** parametricity results carry over,
Takeuti (2001), Bernardy, Jansson and Paterson (2012), Krishnaswami and Dreyer (2013), Atkey, Ghani and Johann (2014)
- Some can be made provable **internally**,
Bernardy, Coquand and Moulin (2015), Guilhem Moulin's PhD (2016)
- Some are **lost**.

Results

- We formulate a **sound** dependent type system ParamDTT.
- We carry over “the” **remaining theorems** metatheoretically.
- We allow proving **additional theorems internally**.

Parametricity

- Variable is **parametric** if **only used for type-checking**
⇒ free well-behavedness theorems.
- Well-studied in System F, System F ω , Haskell, ...

Parametricity in dependent types

- **Some** parametricity results carry over,
Takeuti (2001), Bernardy, Jansson and Paterson (2012), Krishnaswami and Dreyer (2013), Atkey, Ghani and Johann (2014)
- Some can be made provable **internally**,
Bernardy, Coquand and Moulin (2015), Guilhem Moulin's PhD (2016)
- Some are **lost**.

Results

- We formulate a **sound** dependent type system ParamDTT.
- We carry over “the” **remaining theorems** metatheoretically.
- We allow proving **additional theorems internally**.

Parametricity

- Variable is **parametric** if **only used for type-checking**
⇒ free well-behavedness theorems.
- Well-studied in System F, System F ω , Haskell, ...

Parametricity in dependent types

- **Some** parametricity results carry over,
Takeuti (2001), Bernardy, Jansson and Paterson (2012), Krishnaswami and Dreyer (2013), Atkey, Ghani and Johann (2014)
- Some can be made provable **internally**,
Bernardy, Coquand and Moulin (2015), Guilhem Moulin's PhD (2016)
- Some are **lost**.

Results

- We formulate a **sound** dependent type system ParamDTT.
- We carry over “the” **remaining theorems** metatheoretically.
- We allow proving **additional theorems internally**.

Parametricity

- Variable is **parametric** if **only used for type-checking**
⇒ free well-behavedness theorems.
- Well-studied in System F, System F ω , Haskell, ...

Parametricity in dependent types

- **Some** parametricity results carry over,
Takeuti (2001), Bernardy, Jansson and Paterson (2012), Krishnaswami and Dreyer (2013), Atkey, Ghani and Johann (2014)
- Some can be made provable **internally**,
Bernardy, Coquand and Moulin (2015), Guilhem Moulin's PhD (2016)
- Some are **lost**.

Results

- We formulate a **sound** dependent type system ParamDTT.
- We carry over “the” **remaining theorems** metatheoretically.
- We allow proving **additional theorems internally**.

Parametricity

- Variable is **parametric** if **only used for type-checking**
⇒ free well-behavedness theorems.
- Well-studied in System F, System F ω , Haskell, ...

Parametricity in dependent types

- **Some** parametricity results carry over,
Takeuti (2001), Bernardy, Jansson and Paterson (2012), Krishnaswami and Dreyer (2013), Atkey, Ghani and Johann (2014)
- Some can be made provable **internally**,
Bernardy, Coquand and Moulin (2015), Guilhem Moulin's PhD (2016)
- Some are **lost**.

Results

- We formulate a **sound** dependent type system ParamDTT.
- We carry over “the” **remaining theorems** metatheoretically.
- We allow proving **additional theorems internally**.

Parametricity

- Variable is **parametric** if **only used for type-checking**
⇒ free well-behavedness theorems.
- Well-studied in System F, System F ω , Haskell, ...

Parametricity in dependent types

- **Some** parametricity results carry over,
Takeuti (2001), Bernardy, Jansson and Paterson (2012), Krishnaswami and Dreyer (2013), Atkey, Ghani and Johann (2014)
- Some can be made provable **internally**,
Bernardy, Coquand and Moulin (2015), Guilhem Moulin's PhD (2016)
- Some are **lost**.

Results

- We formulate a **sound** dependent type system ParamDTT.
- We carry over “the” **remaining theorems** metatheoretically.
- We allow proving **additional theorems internally**.

Parametricity

- Variable is **parametric** if **only used for type-checking**
⇒ free well-behavedness theorems.
- Well-studied in System F, System F ω , Haskell, ...

Parametricity in dependent types

- **Some** parametricity results carry over,
Takeuti (2001), Bernardy, Jansson and Paterson (2012), Krishnaswami and Dreyer (2013), Atkey, Ghani and Johann (2014)
- Some can be made provable **internally**,
Bernardy, Coquand and Moulin (2015), Guilhem Moulin's PhD (2016)
- Some are **lost**.

Results

- We formulate a **sound** dependent type system ParamDTT.
- We carry over “the” **remaining theorems** metatheoretically.
- We allow proving **additional theorems internally**.

Parametricity, intuitively

In System F, $F\omega$, Haskell, . . . , **type parameters** are parametric.

- Only used for type-checking,
- Not inspected (e.g. no pattern matching),
- Same algorithm on all types.

Enforced by the type system.

Example

`flatten` : $\forall X. \text{Tree } X \rightarrow \text{List } X$

By parametricity:



irrespective of implementation.

Parametricity, intuitively

In System F, $F\omega$, Haskell, . . . , **type parameters** are parametric.

- Only used for type-checking,
- Not inspected (e.g. no pattern matching),
- Same algorithm on all types.

Enforced by the type system.

Example

`flatten` : $\forall X. \text{Tree } X \rightarrow \text{List } X$

By parametricity:

$$\begin{array}{ccccc} A & & \text{Tree } A & \xrightarrow{\text{flatten}} & \text{List } A \\ f \downarrow & & \downarrow \text{Tree } f & & \downarrow \text{List } f \\ B & & \text{Tree } B & \xrightarrow{\text{flatten}} & \text{List } B \end{array}$$

irrespective of implementation.

Theorem

$$(A \rightarrow B) \cong \left(\underbrace{\forall X. (X \rightarrow A)}_{\text{For any representation } (X, r) \text{ of } A} \rightarrow (X \rightarrow B) \right)$$

Proof:

(\rightarrow) $h \mapsto \lambda X. \lambda r. h \circ r$.

(\leftarrow) $g \mapsto g \ A \ \text{id}$.

(src) refl

(tgt) Prove: $g \ X \ r \ x = g \ A \ \text{id} \ (r \ x)$.



Theorem

$$(A \rightarrow B) \cong \left(\underbrace{\forall X. (X \rightarrow A)}_{\text{For any representation } (X, r) \text{ of } A} \rightarrow (X \rightarrow B) \right)$$

Proof:

(\rightarrow) $h \mapsto \lambda X. \lambda r. h \circ r$.

(\leftarrow) $g \mapsto g \ A \ \text{id}$.

(src) refl

(tgt) Prove: $g \ X \ r \ x = g \ A \ \text{id} \ (r \ x)$.



Theorem

$$(A \rightarrow B) \cong \left(\underbrace{\forall X. (X \rightarrow A)}_{\text{For any representation } (X, r) \text{ of } A} \rightarrow (X \rightarrow B) \right)$$

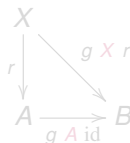
Proof:

(\rightarrow) $h \mapsto \lambda X. \lambda r. h \circ r$.

(\leftarrow) $g \mapsto g \ A \ \text{id}$.

(src) refl

(tgt) Prove: $g \ X \ r \ x = g \ A \ \text{id} \ (r \ x)$.



Theorem

$$(A \rightarrow B) \cong \left(\underbrace{\forall X. (X \rightarrow A)}_{\text{For any representation } (X, r) \text{ of } A} \rightarrow (X \rightarrow B) \right)$$

Proof:

(\rightarrow) $h \mapsto \lambda X. \lambda r. h \circ r$.

(\leftarrow) $g \mapsto g \ A \ \text{id}$.

(src) refl

(tgt) Prove: $g \ X \ r \ x = g \ A \ \text{id} \ (r \ x)$.



Theorem

$$(A \rightarrow B) \cong \left(\underbrace{\forall X. (X \rightarrow A)}_{\text{For any representation } (X, r) \text{ of } A} \rightarrow (X \rightarrow B) \right)$$

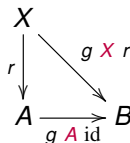
Proof:

(\rightarrow) $h \mapsto \lambda X. \lambda r. h \circ r$.

(\leftarrow) $g \mapsto g \ A \ \text{id}$.

(src) refl

(tgt) Prove: $g \ X \ r \ x = g \ A \ \text{id} \ (r \ x)$.



Lemma

If $g : \forall X. (X \rightarrow A) \rightarrow (X \rightarrow B)$
then $g \ X_0 \ r_0 \ x_0 = g \ A \ \text{id} \ (r_0 \ x_0)$.

Rel. param.: A sound scheme for proving parametricity theorems.
Idea: **Related things map to related things.**

Lemma

If $g : \forall X. (X \rightarrow A) \rightarrow (X \rightarrow B)$
then $g \ X_0 \ r_0 \ x_0 = g \ A \ \text{id} \ (r_0 \ x_0)$.

Rel. param.: A sound scheme for proving parametricity theorems.
Idea: **Related things map to related things.**

Lemma

If $g : \forall X. (X \rightarrow A) \rightarrow (X \rightarrow B)$
then $g \ X_0 \ r_0 \ x_0 = g \ A \ id \ (r_0 \ x_0)$.

Rel. param.: A sound scheme for proving parametricity theorems.
Idea: **Related things map to related things.**

$X : *$, $r : X \rightarrow A$, $x : X \quad \vdash \quad g \ X \ r \ x : B$

Lemma

If $g : \forall X. (X \rightarrow A) \rightarrow (X \rightarrow B)$
then $g \ X_0 \ r_0 \ x_0 = g \ A \ \text{id} \ (r_0 \ x_0)$.

Rel. param.: A sound scheme for proving parametricity theorems.

Idea: **Related things map to related things.**

$X_0 : *$, $r_0 : X_0 \rightarrow A$, $x_0 : X_0$ \vdash $g \ X_0 \ r_0 \ x_0 : B$

$[X] : \text{Rel}$, $[r] : [X \rightarrow A]$, $[x] : [X]$ \vdash

$X_1 : *$, $r_1 : X_1 \rightarrow A$, $x_1 : X_1$ \vdash $g \ X_1 \ r_1 \ x_1 : B$

Lemma

If $g : \forall X. (X \rightarrow A) \rightarrow (X \rightarrow B)$
then $g \ X_0 \ r_0 \ x_0 = g \ A \ \text{id} \ (r_0 \ x_0)$.

Rel. param.: A sound scheme for proving parametricity theorems.

Idea: **Related things map to related things.**

$X_0 : *$, $r_0 : X_0 \rightarrow A$, $x_0 : X_0$ \vdash $g \ X_0 \ r_0 \ x_0 : B$

$[X] : \text{Rel}$, $[r] : [X \rightarrow A]$, $[x] : [X]$ \vdash

$X_1 : *$, $r_1 : X_1 \rightarrow A$, $x_1 : X_1$ \vdash $g \ X_1 \ r_1 \ x_1 : B$

Lemma

If $g : \forall X. (X \rightarrow A) \rightarrow (X \rightarrow B)$
then $g \ X_0 \ r_0 \ x_0 = g \ A \ \text{id} \ (r_0 \ x_0)$.

Rel. param.: A sound scheme for proving parametricity theorems.

Idea: **Related things map to related things.**

$X_0 : *$, $r_0 : X_0 \rightarrow A$, $x_0 : X_0$ \vdash $g \ X_0 \ r_0 \ x_0 : B$

$[X] : \text{Rel}$, $[r] : [X \rightarrow A]$, $[x] : [X]$ \vdash

$X_1 : *$, $r_1 : X_1 \rightarrow A$, $x_1 : X_1$ \vdash $g \ X_1 \ r_1 \ x_1 : B$

Lemma

If $g : \forall X. (X \rightarrow A) \rightarrow (X \rightarrow B)$
then $g \ X_0 \ r_0 \ x_0 = g \ A \ \text{id} \ (r_0 \ x_0)$.

Rel. param.: A sound scheme for proving parametricity theorems.

Idea: **Related things map to related things.**

$X_0 : *$, $r_0 : X_0 \rightarrow A$, $x_0 : X_0$ \vdash $g \ X_0 \ r_0 \ x_0 : B$

$[X] : \text{Rel}$, $[r] : [X \rightarrow A]$, $[x] : [X]$ \vdash

$X_1 : *$, $r_1 : X_1 \rightarrow A$, $x_1 : X_1$ \vdash $g \ X_1 \ r_1 \ x_1 : B$

Lemma

If $g : \forall X. (X \rightarrow A) \rightarrow (X \rightarrow B)$
then $g \ X_0 \ r_0 \ x_0 = g \ A \ \text{id} \ (r_0 \ x_0)$.

Rel. param.: A sound scheme for proving parametricity theorems.

Idea: **Related things map to related things.**

$X_0 : *$, $r_0 : X_0 \rightarrow A$, $x_0 : X_0$ \vdash $g \ X_0 \ r_0 \ x_0 : B$

$[X] : \text{Rel}$, $[r] : [X \rightarrow A]$, $[x] : [X]$ \vdash $[g \ X \ r \ x] : [B]$

$X_1 : *$, $r_1 : X_1 \rightarrow A$, $x_1 : X_1$ \vdash $g \ X_1 \ r_1 \ x_1 : B$

Lemma

If $g : \forall X. (X \rightarrow A) \rightarrow (X \rightarrow B)$
then $g \ X_0 \ r_0 \ x_0 = g \ A \ \text{id} \ (r_0 \ x_0)$.

Rel. param.: A sound scheme for proving parametricity theorems.

Idea: **Related things map to related things.**

$X_0 : *$, $r_0 : X_0 \rightarrow A$, $x_0 : X_0$ \vdash $g \ X_0 \ r_0 \ x_0 : B$

$[X] : \text{Rel}$, $[r] : [X \rightarrow A]$, $[x] : [X]$ \vdash **=**

$X_1 : *$, $r_1 : X_1 \rightarrow A$, $x_1 : X_1$ \vdash $g \ X_1 \ r_1 \ x_1 : B$

IDENTITY EXTENSION LEMMA (IEL)

Lemma

If $g : \forall X. (X \rightarrow A) \rightarrow (X \rightarrow B)$
then $g \ X_0 \ r_0 \ x_0 = g \ A \ \text{id} \ (r_0 \ x_0)$.

Rel. param.: A sound scheme for proving parametricity theorems.

Idea: **Related things map to related things.**

$X_0 : *$, $r_0 : X_0 \rightarrow A$, $x_0 : X_0$ \vdash $g \ X_0 \ r_0 \ x_0 : B$

$[X] : \text{Rel}$, $[r] : [X \rightarrow A]$, $[x] : [X]$ \vdash $=$

$A : *$, $\text{id} : A \rightarrow A$, $r_0 \ x_0 : A$ \vdash $g \ A \ \text{id} \ (r_0 \ x_0) : B$

IDENTITY EXTENSION LEMMA (IEL)

Lemma

If $g : \forall X. (X \rightarrow A) \rightarrow (X \rightarrow B)$
then $g \ X_0 \ r_0 \ x_0 = g \ A \ \text{id} \ (r_0 \ x_0)$.

Rel. param.: A sound scheme for proving parametricity theorems.

Idea: **Related things map to related things.**

$X_0 : *$, $r_0 : X_0 \rightarrow A$, $x_0 : X_0 \vdash g \ X_0 \ r_0 \ x_0 : B$

$/r_0/ : \text{Rel}$, $[r] : \backslash \sqcup \circ r_0 \backslash$, $[x] : /r_0/ \vdash =$

$A : *$, $\text{id} : A \rightarrow A$, $r_0 \ x_0 : A \vdash g \ A \ \text{id} \ (r_0 \ x_0) : B$

IDENTITY EXTENSION LEMMA (IEL)

Lemma

If $g : \forall X. (X \rightarrow A) \rightarrow (X \rightarrow B)$
then $g \ X_0 \ r_0 \ x_0 = g \ A \ \text{id} \ (r_0 \ x_0)$.

Rel. param.: A sound scheme for proving parametricity theorems.

Idea: **Related things map to related things.**

$X_0 : *$, $r_0 : X_0 \rightarrow A$, $x_0 : X_0 \vdash g \ X_0 \ r_0 \ x_0 : B$

$/r_0/ : \text{Rel}$, $\text{refl} : \backslash \sqcup \circ r_0 \backslash$, $\text{refl} : /r_0/ \vdash \quad =$

$A : *$, $\text{id} : A \rightarrow A$, $r_0 \ x_0 : A \vdash g \ A \ \text{id} \ (r_0 \ x_0) : B$

IDENTITY EXTENSION LEMMA (IEL)

This is a **metatheoretical** scheme for System F, System $F\omega$, ...

- Can we do this for **dependent types**?
- Can we do this **internally** in dependent types?

Π is not parametric

System F:

$$\forall X.(X \rightarrow A) \rightarrow (X \rightarrow B).$$

Dependent types:

$$\Pi(X : \mathcal{U}).(X \rightarrow A) \rightarrow (X \rightarrow B).$$

Suppose $B = \mathcal{U}$:

$$\text{leak} : \Pi(X : \mathcal{U}).(X \rightarrow A) \rightarrow (X \rightarrow \mathcal{U})$$

$$\text{leak } X \text{ } r \text{ } x = X.$$

Representation type is returned as data!

In existing work: \mathcal{U} violates identity extension lemma (IEL).

Takeuti (2001), Bernardy, Jansson and Paterson (2012), Krishnaswami and Dreyer (2013), Atkey, Ghani and Johann (2014)

Our solution (syntax-side): Keep track of how we use variables.

Π is not parametric

System F:

$$\forall X.(X \rightarrow A) \rightarrow (X \rightarrow B).$$

Dependent types:

$$\Pi(X : \mathcal{U}).(X \rightarrow A) \rightarrow (X \rightarrow B).$$

Suppose $B = \mathcal{U}$:

$$\text{leak} : \Pi(X : \mathcal{U}).(X \rightarrow A) \rightarrow (X \rightarrow \mathcal{U})$$

$$\text{leak } X \text{ } r \text{ } x = X.$$

Representation type is returned as data!

In existing work: \mathcal{U} violates identity extension lemma (IEL).

Takeuti (2001), Bernardy, Jansson and Paterson (2012), Krishnaswami and Dreyer (2013), Atkey, Ghani and Johann (2014)

Our solution (syntax-side): Keep track of how we use variables.

Π is not parametric

System F:

$$\forall X.(X \rightarrow A) \rightarrow (X \rightarrow B).$$

Dependent types:

$$\Pi(X : \mathcal{U}).(X \rightarrow A) \rightarrow (X \rightarrow B).$$

Suppose $B = \mathcal{U}$:

$$\text{leak} : \Pi(X : \mathcal{U}).(X \rightarrow A) \rightarrow (X \rightarrow \mathcal{U})$$

$$\text{leak } X \ r \ x = X.$$

Representation type is returned as data!

In existing work: \mathcal{U} violates identity extension lemma (IEL).

Takeuti (2001), Bernardy, Jansson and Paterson (2012), Krishnaswami and Dreyer (2013), Atkey, Ghani and Johann (2014)

Our solution (syntax-side): Keep track of how we use variables.

Π is not parametric

System F:

$$\forall X.(X \rightarrow A) \rightarrow (X \rightarrow B).$$

Dependent types:

$$\Pi(X : \mathcal{U}).(X \rightarrow A) \rightarrow (X \rightarrow B).$$

Suppose $B = \mathcal{U}$:

$$\text{leak} : \Pi(X : \mathcal{U}).(X \rightarrow A) \rightarrow (X \rightarrow \mathcal{U})$$

$$\text{leak } X \ r \ x = X.$$

Representation type is returned as data!

In existing work: \mathcal{U} violates identity extension lemma (IEL).

Takeuti (2001), Bernardy, Jansson and Paterson (2012), Krishnaswami and Dreyer (2013), Atkey, Ghani and Johann (2014)

Our solution (syntax-side): Keep track of how we use variables.

Π is not parametric

System F:

$$\forall X.(X \rightarrow A) \rightarrow (X \rightarrow B).$$

Dependent types:

$$\Pi(X : \mathcal{U}).(X \rightarrow A) \rightarrow (X \rightarrow B).$$

Suppose $B = \mathcal{U}$:

$$\text{leak} : \Pi(X : \mathcal{U}).(X \rightarrow A) \rightarrow (X \rightarrow \mathcal{U})$$

$$\text{leak } X \ r \ x = X.$$

Representation type is returned as data!

In existing work: \mathcal{U} violates identity extension lemma (IEL).

Takeuti (2001), Bernardy, Jansson and Paterson (2012), Krishnaswami and Dreyer (2013), Atkey, Ghani and Johann (2014)

Our solution (syntax-side): Keep track of how we use variables.

Adding parametric quantifiers

Non-parametric quantifiers

Non-parametric functions

$$f : \prod(x : A).B x, \quad f : A \rightarrow B$$

can use argument **as data**.

$$\frac{A : \mathcal{U} \quad B : A \rightarrow \mathcal{U}}{\prod(x : A).B x : \mathcal{U}}$$

Non-parametric pairs

$$p : \sum(x : A).B x, \quad p : A \times B$$

Parametric quantifiers

Parametric functions

$$f : \forall(x : A).B x$$

cannot inspect argument: it is used **only for type-checking**.*

$$\frac{A : \mathcal{U} \quad B : A \rightarrow \mathcal{U}}{\forall(x : A).B x : \mathcal{U}}$$

Parametric pairs (packs)

$$p : \exists(x : A).B x$$

*if A satisfies certain properties.

Adding parametric quantifiers

Non-parametric quantifiers

Non-parametric functions

$$f : \prod(x : A).B x, \quad f : A \rightarrow B$$

can use argument **as data**.

$$\frac{A : \mathcal{U} \quad B : A \rightarrow \mathcal{U}}{\prod(x : A).B x : \mathcal{U}}$$

Non-parametric pairs

$$p : \sum(x : A).B x, \quad p : A \times B$$

Parametric quantifiers

Parametric functions

$$f : \forall(x : A).B x$$

cannot inspect argument: it is used **only for type-checking**.*

$$\frac{A : \mathcal{U} \quad B : A \rightarrow \mathcal{U}}{\forall(x : A).B x : \mathcal{U}}$$

Parametric pairs (packs)

$$p : \exists(x : A).B x$$

*if A satisfies certain properties.

Adding parametric quantifiers

Non-parametric quantifiers

Non-parametric functions

$$f : \prod(x : A).B x, \quad f : A \rightarrow B$$

can use argument **as data**.

$$\frac{A : \mathcal{U} \quad B : A \rightarrow \mathcal{U}}{\prod(x : A).B x : \mathcal{U}}$$

Non-parametric pairs

$$p : \sum(x : A).B x, \quad p : A \times B$$

Parametric quantifiers

Parametric functions

$$f : \forall(x : A).B x$$

cannot inspect argument: it is used **only for type-checking**.*

$$\frac{A : \mathcal{U} \quad B : A \rightarrow \mathcal{U}}{\forall(x : A).B x : \mathcal{U}}$$

Parametric pairs (packs)

$$p : \exists(x : A).B x$$

*if A satisfies certain properties.

Adding parametric quantifiers

Non-parametric quantifiers

Non-parametric functions

$$f : \prod(x : A).B x, \quad f : A \rightarrow B$$

can use argument **as data**.

$$\frac{A : \mathcal{U} \quad B : A \rightarrow \mathcal{U}}{\prod(x : A).B x : \mathcal{U}}$$

Non-parametric pairs

$$p : \sum(x : A).B x, \quad p : A \times B$$

Parametric quantifiers

Parametric functions

$$f : \forall(x : A).B x$$

cannot inspect argument: it is used **only for type-checking**.*

$$\frac{A : \mathcal{U} \quad B : A \rightarrow \mathcal{U}}{\forall(x : A).B x : \mathcal{U}}$$

Parametric pairs (packs)

$$p : \exists(x : A).B x$$

*if A satisfies certain properties.

Use \forall if you want representation independence:

$$\forall(X : \mathcal{U}).(X \rightarrow A) \rightarrow (X \rightarrow B).$$

The following is now ill-typed:

$$\textit{leak} : \forall(X : \mathcal{U}).(X \rightarrow A) \rightarrow (X \rightarrow \mathcal{U})$$

$$\textit{leak } X \textit{ r } x = X,$$

since we cannot use X as data.

Use \forall if you want representation independence:

$$\forall(X : \mathcal{U}).(X \rightarrow A) \rightarrow (X \rightarrow B).$$

The following is now ill-typed:

$$\text{leak} : \forall(X : \mathcal{U}).(X \rightarrow A) \rightarrow (X \rightarrow \mathcal{U})$$

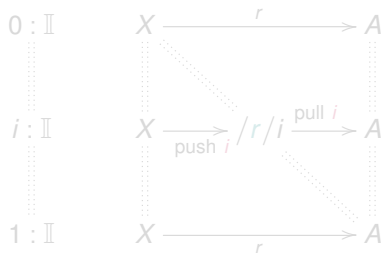
$$\text{leak } X \ r \ x = X,$$

since we cannot use X as data.

An internal proof

Tools:

- Relational interval “type”:
 $0 \sim 1 : \mathbb{I}$ (cf. Bernardy, Coquand and Moulin (2015), and cubical TT)
- Graph type of $r : X \rightarrow A$
 $/r/ : \mathbb{I} \rightarrow \mathcal{U}$
 $/r/0 = X$
 $/r/1 = A$
(not a primitive)



Lemma

If $g : \forall X. (X \rightarrow A) \rightarrow (X \rightarrow B)$
then $g \ X \ r \ x =_B g \ A \ \text{id} \ (r \ x)$.

Proof:

$p : \forall (_ : \mathbb{I}). B$

$p \ i = g \ (/r/i) \ (\text{pull } i) \ (\text{push } i \ x)$

$p \ 0 = g \ X \ r \ (\text{id } x)$

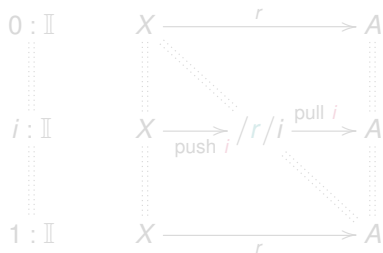
$p \ 1 = g \ A \ \text{id} \ (r \ x)$

IEL: p is constant. □

An internal proof

Tools:

- Relational interval “type”:
 $0 \curvearrowright 1 : \mathbb{I}$ (cf. Bernardy, Coquand and Moulin (2015), and cubical TT)
- Graph type of $r : X \rightarrow A$
 $/r/ : \mathbb{I} \rightarrow \mathcal{U}$
 $/r/0 = X$
 $/r/1 = A$
(not a primitive)



Lemma

If $g : \forall X. (X \rightarrow A) \rightarrow (X \rightarrow B)$
then $g X r x =_B g A \text{id } (r x)$.

Proof:

$p : \forall (- : \mathbb{I}). B$

$p i = g (/r/i) (\text{pull } i) (\text{push } i x)$

$p 0 = g X r (\text{id } x)$

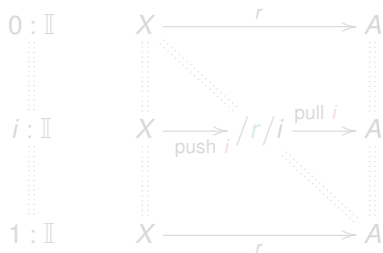
$p 1 = g A \text{id } (r x)$

IEL: p is constant. □

An internal proof

Tools:

- Relational interval “type”:
 $0 \curvearrowright 1 : \mathbb{I}$ (cf. Bernardy, Coquand and Moulin (2015), and cubical TT)
- Graph type of $r : X \rightarrow A$
 $/r/ : \mathbb{I} \rightarrow \mathcal{U}$
 $/r/0 = X$
 $/r/1 = A$
(not a primitive)



Lemma

If $g : \forall X. (X \rightarrow A) \rightarrow (X \rightarrow B)$
then $g \ X \ r \ x =_B g \ A \ \text{id} \ (r \ x)$.

Proof:

$p : \forall (- : \mathbb{I}). B$

$p \ i = g \ (/r/i) \ (\text{pull } i) \ (\text{push } i \ x)$

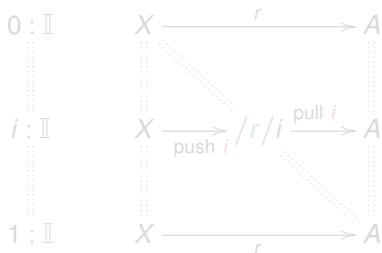
$p \ 0 = g \ X \ r \ (\text{id } x)$

$p \ 1 = g \ A \ \text{id} \ (r \ x)$

IEL: p is constant. □

Tools:

- Relational interval “type”:
 $0 \curvearrowright 1 : \mathbb{I}$ (cf. Bernardy, Coquand and Moulin (2015), and cubical TT)
- Graph type of $r : X \rightarrow A$
 $/r/ : \mathbb{I} \rightarrow \mathcal{U}$
 $/r/0 = X$
 $/r/1 = A$
(not a primitive)



Lemma

If $g : \forall X. (X \rightarrow A) \rightarrow (X \rightarrow B)$
then $g \ X \ r \ x =_B g \ A \ \text{id} \ (r \ x)$.

Proof:

$p : \forall (- : \mathbb{I}). B$

$p \ i = g \ (/r/i) \ (\text{pull } i) \ (\text{push } i \ x)$

$p \ 0 = g \ X \ r \ (\text{id } x)$

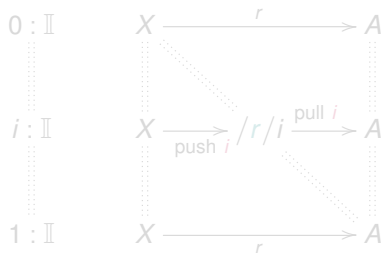
$p \ 1 = g \ A \ \text{id} \ (r \ x)$

IEL: p is constant. □

An internal proof

Tools:

- Relational interval “type”:
 $0 \curvearrowright 1 : \mathbb{I}$ (cf. Bernardy, Coquand and Moulin (2015), and cubical TT)
- Graph type of $r : X \rightarrow A$
 $/r/ : \mathbb{I} \rightarrow \mathcal{U}$
 $/r/0 = X$
 $/r/1 = A$
(not a primitive)



Lemma

If $g : \forall X. (X \rightarrow A) \rightarrow (X \rightarrow B)$
then $g \ X \ r \ x =_B g \ A \ \text{id} \ (r \ x)$.

Proof:

$p : \forall (- : \mathbb{I}). B$

$p \ i = g \ (/r/i) \ (\text{pull } i) \ (\text{push } i \ x)$

$p \ 0 = g \ X \ r \ (\text{id } x)$

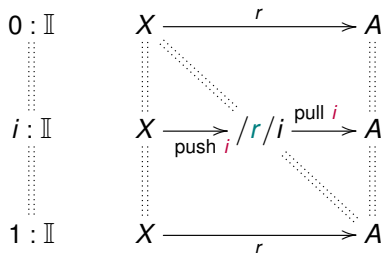
$p \ 1 = g \ A \ \text{id} \ (r \ x)$

IEL: p is constant. □

An internal proof

Tools:

- Relational interval “type”:
 $0 \curvearrowright 1 : \mathbb{I}$ (cf. Bernardy, Coquand and Moulin (2015), and cubical TT)
- Graph type of $r : X \rightarrow A$
 $/r/ : \mathbb{I} \rightarrow \mathcal{U}$
 $/r/0 = X$
 $/r/1 = A$
 (not a primitive)



Lemma

If $g : \forall X. (X \rightarrow A) \rightarrow (X \rightarrow B)$
 then $g X r x =_B g A \text{id } (r x)$.

Proof:

$p : \forall (- : \mathbb{I}). B$

$p i = g (/r/i) (\text{pull } i) (\text{push } i x)$

$p 0 = g X r (\text{id } x)$

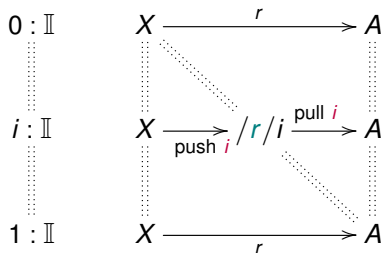
$p 1 = g A \text{id } (r x)$

IEL: p is constant. □

An internal proof

Tools:

- Relational interval “type”:
 $0 \curvearrowright 1 : \mathbb{I}$ (cf. Bernardy, Coquand and Moulin (2015), and cubical TT)
- Graph type of $r : X \rightarrow A$
 $/r/ : \mathbb{I} \rightarrow \mathcal{U}$
 $/r/0 = X$
 $/r/1 = A$
 (not a primitive)



Lemma

If $g : \forall X. (X \rightarrow A) \rightarrow (X \rightarrow B)$
 then $g \ X \ r \ x =_B g \ A \ \text{id} \ (r \ x)$.

Proof:

$p : \forall (- : \mathbb{I}). B$

$p \ i = g \ (/r/i) \ (\text{pull } i) \ (\text{push } i \ x)$

$p \ 0 = g \ X \ r \ (\text{id } x)$

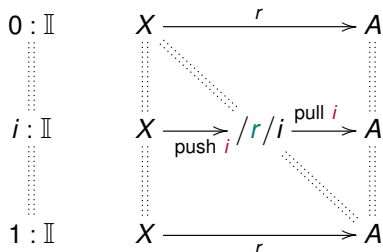
$p \ 1 = g \ A \ \text{id} \ (r \ x)$

IEL: p is constant. □

An internal proof

Tools:

- Relational interval “type”:
 $0 \curvearrowright 1 : \mathbb{I}$ (cf. Bernardy, Coquand and Moulin (2015), and cubical TT)
- Graph type of $r : X \rightarrow A$
 $/r/ : \mathbb{I} \rightarrow \mathcal{U}$
 $/r/0 = X$
 $/r/1 = A$
 (not a primitive)



Lemma

If $g : \forall X. (X \rightarrow A) \rightarrow (X \rightarrow B)$
 then $g \ X \ r \ x =_B g \ A \ \text{id} \ (r \ x)$.

Proof:

$p : \forall (- : \mathbb{I}). B$

$p \ i = g \ (/r/i)$ (pull i) (push $i \ x$)

$p \ 0 = g \ X \ r \ (\text{id} \ x)$

$p \ 1 = g \ A \ \text{id} \ (r \ x)$

IEL: p is constant. □

The framework:

- Type system **ParamDTT** with Π and Σ , \forall and \exists ,
- **Soundness** using ‘bridge/path cubical sets’ (higher-dimensional labelled reflexive graphs),
- We **extend Agda** with support for ParamDTT,

Results:

- **Stronger internal** parametricity system,
 - but not fully iterated,
- We show **internally** that **Church encoding** of data (e.g. lists) and codata (e.g. streams) works,
 - up to predicativity issues,
- We support **sized types**:
 - modular, type-directed approach to termination checking,
 - requires **dependent parametric** quantification.

The framework:

- Type system **ParamDTT** with Π and Σ , \forall and \exists ,
- **Soundness** using ‘bridge/path cubical sets’ (higher-dimensional labelled reflexive graphs),
- We **extend Agda** with support for ParamDTT,

Results:

- **Stronger internal** parametricity system,
 - but not fully iterated,
- We show **internally** that **Church encoding** of data (e.g. lists) and codata (e.g. streams) works,
 - up to predicativity issues,
- We support **sized types**:
 - modular, type-directed approach to termination checking,
 - requires **dependent parametric** quantification.

The framework:

- Type system **ParamDTT** with Π and Σ , \forall and \exists ,
- **Soundness** using ‘bridge/path cubical sets’ (higher-dimensional labelled reflexive graphs),
- We **extend Agda** with support for ParamDTT,

Results:

- **Stronger internal** parametricity system,
 - but not fully iterated,
- We show **internally** that **Church encoding** of data (e.g. lists) and codata (e.g. streams) works,
 - up to predicativity issues,
- We support **sized types**:
 - modular, type-directed approach to termination checking,
 - requires **dependent parametric** quantification.

The framework:

- Type system **ParamDTT** with Π and Σ , \forall and \exists ,
- **Soundness** using ‘bridge/path cubical sets’ (higher-dimensional labelled reflexive graphs),
- We **extend Agda** with support for ParamDTT,

Results:

- **Stronger internal** parametricity system,
 - but not fully iterated,
- We show **internally** that **Church encoding** of data (e.g. lists) and codata (e.g. streams) works,
 - up to predicativity issues,
- We support **sized types**:
 - modular, type-directed approach to termination checking,
 - requires **dependent parametric** quantification.

The framework:

- Type system **ParamDTT** with Π and Σ , \forall and \exists ,
- **Soundness** using ‘bridge/path cubical sets’ (higher-dimensional labelled reflexive graphs),
- We **extend Agda** with support for ParamDTT,

Results:

- **Stronger internal** parametricity system,
 - but not fully iterated,
- We show **internally** that **Church encoding** of data (e.g. lists) and codata (e.g. streams) works,
 - up to predicativity issues,
- We support **sized types**:
 - modular, type-directed approach to termination checking,
 - requires **dependent parametric** quantification.

The framework:

- Type system **ParamDTT** with Π and Σ , \forall and \exists ,
- **Soundness** using ‘bridge/path cubical sets’ (higher-dimensional labelled reflexive graphs),
- We **extend Agda** with support for ParamDTT,

Results:

- **Stronger internal** parametricity system,
 - but not fully iterated,
- We show **internally** that **Church encoding** of data (e.g. lists) and codata (e.g. streams) works,
 - up to predicativity issues,
- We support **sized types**:
 - modular, type-directed approach to termination checking,
 - requires **dependent parametric** quantification.

Thanks!

Questions?

Parametricity, Shape-irrelevance, irrelevance

4 functors on bridge/path cubical sets:

- id Non-parametricity (continuity)
- ‡ Parametricity
- Shape-irrelevance
 - Irrelevance

such that $\ddagger \circ (\bullet\bullet) = \bullet$.

Abel has: $\llbracket \text{Size} \rrbracket = \llbracket \mathbb{N} \rrbracket$.

We have: $\llbracket \text{Size} \rrbracket = \bullet\bullet \llbracket \mathbb{N} \rrbracket$. Hence, $\ddagger \llbracket \text{Size} \rrbracket = \bullet \llbracket \mathbb{N} \rrbracket$.

ParamDTT	Abel	$\llbracket \text{domain} \rrbracket$
$\prod (i : \text{Size}). A i$	$\bullet\bullet (i : \text{Size}) \rightarrow A i$	$\bullet\bullet \llbracket \mathbb{N} \rrbracket$
$\forall (i : \text{Size}). A i$	$\bullet (i : \text{Size}) \rightarrow A i$	$\bullet \llbracket \mathbb{N} \rrbracket$

Church initial algebras

Assume level-preserving functor F .

$$\text{Mu}_\ell = \forall (X : \mathcal{U}_\ell). (FX \rightarrow X) \rightarrow X.$$

$$\text{mkMu}_\ell : F\text{Mu}_\ell \rightarrow \text{Mu}_\ell.$$

$$\text{fold}_\ell A \text{ mkA} = \lambda m. (m A \text{ mkA}) : \text{Mu}_\ell \rightarrow A$$

$$\downarrow_\ell = \lambda m. m|_{\mathcal{U}_\ell} : \text{Mu}_{\ell+1} \rightarrow \text{Mu}_\ell$$

Theorem (Initiality of Mu up to \downarrow)

For any B , $\text{mk}B$ and any algebra morphism $f : \text{Mu}_\ell \rightarrow B$:

$$\begin{array}{ccc} \text{Mu}_{\ell+1} & & \\ \downarrow \downarrow_\ell & \searrow \text{fold}_{\ell+1} B \text{ mk}B & \\ \text{Mu}_\ell & \xrightarrow{f} & B \end{array}$$

Church initial algebras

Assume level-preserving functor F .

$$\text{Mu}_\ell = \forall (X : \mathcal{U}_\ell). (FX \rightarrow X) \rightarrow X.$$

$$\text{mkMu}_\ell : F\text{Mu}_\ell \rightarrow \text{Mu}_\ell.$$

$$\text{fold}_\ell \mathbf{A} \text{ mkA} = \lambda m. (m \mathbf{A} \text{ mkA}) : \text{Mu}_\ell \rightarrow \mathbf{A}$$

$$\downarrow_\ell = \lambda m. m|_{\mathcal{U}_\ell} : \text{Mu}_{\ell+1} \rightarrow \text{Mu}_\ell$$

Theorem (Initiality of Mu up to \downarrow)

For any B , $\text{mk}B$ and any algebra morphism $f : \text{Mu}_\ell \rightarrow B$:

$$\begin{array}{ccc} \text{Mu}_{\ell+1} & & \\ \downarrow \downarrow_\ell & \searrow \text{fold}_{\ell+1} \mathbf{B} \text{ mk}B & \\ \text{Mu}_\ell & \xrightarrow{f} & B \end{array}$$

Church initial algebras

Assume level-preserving functor F .

$$\text{Mu}_\ell = \forall (X : \mathcal{U}_\ell). (FX \rightarrow X) \rightarrow X.$$

$$\text{mkMu}_\ell : F\text{Mu}_\ell \rightarrow \text{Mu}_\ell.$$

$$\text{fold}_\ell \mathbf{A} \text{ mkA} = \lambda m. (m \mathbf{A} \text{ mkA}) : \text{Mu}_\ell \rightarrow \mathbf{A}$$

$$\downarrow_\ell = \lambda m. m|_{\mathcal{U}_\ell} : \text{Mu}_{\ell+1} \rightarrow \text{Mu}_\ell$$

Theorem (Initiality of Mu up to \downarrow)

For any B , $\text{mk}B$ and any algebra morphism $f : \text{Mu}_\ell \rightarrow B$:

$$\begin{array}{ccc} \text{Mu}_{\ell+1} & & \\ \downarrow \downarrow_\ell & \searrow \text{fold}_{\ell+1} \mathbf{B} \text{ mkB} & \\ \text{Mu}_\ell & \xrightarrow{f} & B \end{array}$$

Church initial algebras

Assume level-preserving functor F .

$$\text{Mu}_\ell = \forall (X : \mathcal{U}_\ell). (FX \rightarrow X) \rightarrow X.$$

$$\text{mkMu}_\ell : F\text{Mu}_\ell \rightarrow \text{Mu}_\ell.$$

$$\text{fold}_\ell \ A \ \text{mk}A = \lambda m. (m \ A \ \text{mk}A) : \text{Mu}_\ell \rightarrow A$$

$$\downarrow_\ell = \lambda m. m|_{\mathcal{U}_\ell} : \text{Mu}_{\ell+1} \rightarrow \text{Mu}_\ell$$

Theorem (Initiality of Mu up to \downarrow)

For any B , $\text{mk}B$ and any algebra morphism $f : \text{Mu}_\ell \rightarrow B$:

$$\begin{array}{ccc} \text{Mu}_{\ell+1} & & \\ \downarrow_{\downarrow_\ell} \swarrow_{\text{fold}_{\ell+1} \ B \ \text{mk}B} & & \\ \text{Mu}_\ell & \xrightarrow{f} & B \end{array}$$

To build a fixpoint $\text{List } A$ of $(\text{Unit} + A \times \sqcup)$:

- By well-founded induction on $n : \text{Size}$, build $\widehat{\text{List}} A n \cong \text{Unit} + A \times (\exists m < n. \widehat{\text{List}} A m)$,
 - Special fixpoint operator for Size .
- $\text{List } A := \exists n. \widehat{\text{List}} A n$.

Parametricity: side bounds are hidden

Works for finitely branching container functors (even indexed):

$$F T = \Sigma(c : C).(B c \rightarrow T).$$

Also final co-algebras (e.g. streams).

To build a fixpoint $\text{List } A$ of $(\text{Unit} + A \times \sqcup)$:

- By well-founded induction on $n : \text{Size}$, build $\widehat{\text{List } A} n \cong \text{Unit} + A \times (\exists m < n. \widehat{\text{List } A} m)$,
 - Special fixpoint operator for Size .
- $\text{List } A := \exists n. \widehat{\text{List } A} n$.

Parametricity: side bounds are hidden

Works for finitely branching container functors (even indexed):

$F T = \Sigma(c : C). (B c \rightarrow T)$.

Also final co-algebras (e.g. streams).

To build a fixpoint $\text{List } A$ of $(\text{Unit} + A \times \sqcup)$:

- By well-founded induction on $n : \text{Size}$, build $\widehat{\text{List } A} n \cong \text{Unit} + A \times (\exists m < n. \widehat{\text{List } A} m)$,
 - Special fixpoint operator for Size .
- $\text{List } A := \exists n. \widehat{\text{List } A} n$.

Parametricity: side bounds are hidden

Works for finitely branching container functors (even indexed):

$F T = \Sigma(c : C). (B c \rightarrow T)$.

Also final co-algebras (e.g. streams).

To build a fixpoint $\text{List } A$ of $(\text{Unit} + A \times \sqcup)$:

- By well-founded induction on $n : \text{Size}$, build $\widehat{\text{List}} A n \cong \text{Unit} + A \times (\exists m < n. \widehat{\text{List}} A m)$,
 - Special fixpoint operator for Size .
- $\text{List } A := \exists n. \widehat{\text{List}} A n$.

Parametricity: side bounds are hidden

Works for finitely branching container functors (even indexed):

$F T = \Sigma(c : C). (B c \rightarrow T)$.

Also final co-algebras (e.g. streams).

To build a fixpoint $\text{List } A$ of $(\text{Unit} + A \times \sqcup)$:

- By well-founded induction on $n : \text{Size}$, build $\widehat{\text{List}} A n \cong \text{Unit} + A \times (\exists m < n. \widehat{\text{List}} A m)$,
 - Special fixpoint operator for Size .
- $\text{List } A := \exists n. \widehat{\text{List}} A n$.

Parametricity: side bounds are hidden

Works for finitely branching container functors (even indexed):

$$F T = \Sigma(c : C).(B c \rightarrow T).$$

Also final co-algebras (e.g. streams).

To build a fixpoint List A of $(\text{Unit} + A \times \sqcup)$:

- By well-founded induction on $n : \text{Size}$, build $\widehat{\text{List}} A n \cong \text{Unit} + A \times (\exists m < n. \widehat{\text{List}} A m)$,
 - Special fixpoint operator for Size.
- $\text{List } A := \exists n. \widehat{\text{List}} A n$.

Parametricity: side bounds are hidden

Works for finitely branching container functors (even indexed):

$$F T = \Sigma(c : C).(B c \rightarrow T).$$

Also final co-algebras (e.g. streams).

Example of a bridge/path cubical set

The context $(i : \prod \mathbb{I}, j : \forall \mathbb{I})$ as a bridge/path cubical set.

legend: $\bullet \xrightarrow{\text{bridge}} \bullet$ $\bullet \xlongequal{\text{path}} \bullet$

