Parametric Quantifiers for Dependent Type Theory

Andreas Nuyts*, Andrea Vezzosi‡, Dominique Devriese*

*KU Leuven, ‡Chalmers University of Technology

ICFP 2017
Oxford, UK
September 6, 2017

Overview

Parametricity

- Type variable is **parametric** if **only used for type-checking** $\Rightarrow$ free well-behavedness theorems.
- Well-studied in System F, System Fω, 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**.

Overview

Parametricity

- Type variable is **parametric** if **only used for type-checking** $\Rightarrow$ free well-behavedness theorems.
- Well-studied in System F, System Fω, 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**.

Overview

Parametricity

- Type variable is **parametric** if **only used for type-checking** $\Rightarrow$ free well-behavedness theorems.
- Well-studied in System F, System Fω, 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

- Type variable is **parametric** if **only used for type-checking** $\Rightarrow$ free well-behavedness theorems.
- Well-studied in System F, System Fω, 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**.

Overview

Parametricity

- Type variable is **parametric** if **only used for type-checking** $\Rightarrow$ free well-behavedness theorems.
- Well-studied in System F, System Fω, 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**.

Overview

Parametricity

- Type variable is **parametric** if **only used for type-checking** $\Rightarrow$ free well-behavedness theorems.
- Well-studied in System F, System Fω, 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**.

Overview

Parametricity

- Type variable is **parametric** if **only used for type-checking** $\Rightarrow$ free well-behavedness theorems.
- Well-studied in System F, System Fω, 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

- Type variable is **parametric** if **only used for type-checking** $\Rightarrow$ free well-behavedness theorems.
- Well-studied in System F, System Fω, 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**.

Overview

Parametricity

- Type variable is **parametric** if **only used for type-checking** $\Rightarrow$ free well-behavedness theorems.
- Well-studied in System F, System Fω, 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, Fω, 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

$$\text{flatten} \quad : \quad \forall X.\text{Tree } X \to \text{List } X$$

By parametricity:

$$\begin{tikzcd} A \arrow[d, "f"'] & \text{Tree } A \arrow[r, "\text{flatten}"] \arrow[d, "\text{Tree } f"'] & \text{List } A \arrow[d, "\text{List } f"] \\ B & \text{Tree } B \arrow[r, "\text{flatten}"] & \text{List } B \end{tikzcd}$$

irrespective of implementation.

Parametricity, intuitively

In System F, Fω, 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

$$\text{flatten} \quad : \quad \forall X.\text{Tree } X \to \text{List } X$$

By parametricity:

$$\begin{tikzcd} A \arrow[d, "f"'] & \text{Tree } A \arrow[r, "\text{flatten}"] \arrow[d, "\text{Tree } f"'] & \text{List } A \arrow[d, "\text{List } f"] \\ B & \text{Tree } B \arrow[r, "\text{flatten}"] & \text{List } B \end{tikzcd}$$

irrespective of implementation.

Parametricity: Representation independence

Theorem

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

Proof:

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

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

(src) refl

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

$$\begin{tikzcd} X \arrow[d, "r"'] \arrow[dr, "g\ X\ r"] & \\ A \arrow[r, "g\ A\ \mathrm{id}"'] & B \end{tikzcd}$$

Parametricity: Representation independence

Theorem

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

Proof:

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

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

(src) refl

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

$$\begin{tikzcd} X \arrow[d, "r"'] \arrow[dr, "g\ X\ r"] & \\ A \arrow[r, "g\ A\ \mathrm{id}"'] & B \end{tikzcd}$$

Theorem

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

Proof:

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

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

(src) refl

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

$$\begin{tikzcd}
X \arrow[d, "r"'] \arrow[dr, "g\ X\ r"] & \\
A \arrow[r, "g\ A\ \mathrm{id}"'] & B
\end{tikzcd}$$

Theorem

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

Proof:

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

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

(src) refl

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

$$\begin{tikzcd} X \arrow[d, "r"'] \arrow[dr, "g\ X\ r"] & \\ A \arrow[r, "g\ A\ \mathrm{id}"'] & B \end{tikzcd}$$

Parametricity: Representation independence

Theorem

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

Proof:

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

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

(src) refl

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

$$\begin{tikzcd}
X \arrow[d, "r"'] \arrow[dr, "g\ X\ r"] & \\
A \arrow[r, "g\ A\ \mathrm{id}"'] & B
\end{tikzcd}$$

Relational parametricity

Lemma

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

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

Relational parametricity

Lemma

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

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

Relational parametricity

Lemma

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

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

$$X : *, \qquad r : X \to A, \qquad x : X \qquad \vdash \qquad g\ X\ r\ x : B$$

Lemma

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

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

$$
\begin{array}{lllll}
X_0 : *, & r_0 : X_0 \to A, & x_0 : X_0 & \vdash & g\ X_0\ r_0\ x_0 : B \\
[X] : \mathrm{Rel}, & [r] : [X \to A], & [x] : [X] & \vdash & \\
X_1 : *, & r_1 : X_1 \to A, & x_1 : X_1 & \vdash & g\ X_1\ r_1\ x_1 : B
\end{array}
$$

Relational parametricity

Lemma

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

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

$$
\begin{array}{lllll}
X_0 : *, & r_0 : X_0 \to A, & x_0 : X_0 & \vdash & g\ X_0\ r_0\ x_0 : B \\
[X] : \mathrm{Rel}, & [r] : [X \to A], & [x] : [X] & \vdash & \\
X_1 : *, & r_1 : X_1 \to A, & x_1 : X_1 & \vdash & g\ X_1\ r_1\ x_1 : B
\end{array}
$$

Relational parametricity

Lemma

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

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

$$
\begin{array}{ccccc}
X_0 : *, & r_0 : X_0 \to A, & x_0 : X_0 & \vdash & g\ X_0\ r_0\ x_0 : B \\
[X] : \mathrm{Rel}, & [r] : [X \to A], & [x] : [X] & \vdash & \\
X_1 : *, & r_1 : X_1 \to A, & x_1 : X_1 & \vdash & g\ X_1\ r_1\ x_1 : B
\end{array}
$$

Relational parametricity

Lemma

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

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

$$\begin{array}{ccccc}
X_0 : *, & r_0 : X_0 \to A, & x_0 : X_0 & \vdash & g\ X_0\ r_0\ x_0 : B \\
[X] : \mathrm{Rel}, & [r] : [X \to A], & [x] : [X] & \vdash & \\
X_1 : *, & r_1 : X_1 \to A, & x_1 : X_1 & \vdash & g\ X_1\ r_1\ x_1 : B
\end{array}$$

Relational parametricity

Lemma

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

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

$$\begin{array}{ccccc} X_0 : *, & r_0 : X_0 \to A, & x_0 : X_0 & \vdash & g\ X_0\ r_0\ x_0 : B \\ [X] : \mathrm{Rel}, & [r] : [X \to A], & [x] : [X] & \vdash & [g\ X\ r\ x] : [B] \\ X_1 : *, & r_1 : X_1 \to A, & x_1 : X_1 & \vdash & g\ X_1\ r_1\ x_1 : B \end{array}$$

Lemma

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

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

$$
\begin{array}{ccccc}
X_0 : *, & r_0 : X_0 \to A, & x_0 : X_0 & \vdash & g\ X_0\ r_0\ x_0 : B \\
[X] : \mathrm{Rel}, & [r] : [X \to A], & [x] : [X] & \vdash & = \\
X_1 : *, & r_1 : X_1 \to A, & x_1 : X_1 & \vdash & g\ X_1\ r_1\ x_1 : B
\end{array}
$$

IDENTITY EXTENSION LEMMA (IEL)

Lemma

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

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

$$
\begin{array}{ccccc}
X_0 : *, & r_0 : X_0 \to A, & x_0 : X_0 & \vdash & g\ X_0\ r_0\ x_0 : B \\
[X] : \mathrm{Rel}, & [r] : [X \to A], & [x] : [X] & \vdash & = \\
A : *, & \mathrm{id} : A \to A, & r_0\ x_0 : A & \vdash & g\ A\ \mathrm{id}\ (r_0\ x_0) : B
\end{array}
$$

IDENTITY EXTENSION LEMMA (IEL)

Lemma

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

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

$$
\begin{array}{ccccc}
X_0 : *, & r_0 : X_0 \to A, & x_0 : X_0 & \vdash & g\ X_0\ r_0\ x_0 : B \\
/r_0/ : \mathrm{Rel}, & [r] : \backslash \sqcup \circ r_0 \backslash, & [x] : /r_0/ & \vdash & = \\
A : *, & \mathrm{id} : A \to A, & r_0\ x_0 : A & \vdash & g\ A\ \mathrm{id}\ (r_0\ x_0) : B
\end{array}
$$

IDENTITY EXTENSION LEMMA (IEL)

Lemma

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

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

$$
\begin{array}{ccccc}
X_0 : *, & r_0 : X_0 \to A, & x_0 : X_0 & \vdash & g\ X_0\ r_0\ x_0 : B \\
/r_0/ : \mathrm{Rel}, & \mathrm{refl} : \backslash \sqcup \circ r_0 \backslash, & \mathrm{refl} : /r_0/ & \vdash & = \\
A : *, & \mathrm{id} : A \to A, & r_0\ x_0 : A & \vdash & g\ A\ \mathrm{id}\ (r_0\ x_0) : B
\end{array}
$$

IDENTITY EXTENSION LEMMA (IEL)

Relational parametricity

This is a **metatheoretical** scheme for System F, System Fω, ...

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

Π is not parametric

System F:
$\forall X.(X \to A) \to (X \to B)$.

Dependent types:
$\Pi(X : \mathcal{U}).(X \to A) \to (X \to B)$.

Suppose $B = \mathcal{U}$:
$leak : \Pi(X : \mathcal{U}).(X \to A) \to (X \to \mathcal{U})$
$leak\ X\ r\ x = X$.
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:
$\forall X.(X \to A) \to (X \to B)$.

Dependent types:
$\Pi(X : \mathcal{U}).(X \to A) \to (X \to B)$.

Suppose $B = \mathcal{U}$:
$leak : \Pi(X : \mathcal{U}).(X \to A) \to (X \to \mathcal{U})$
$leak\ X\ r\ x = X$.
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:
$\forall X.(X \to A) \to (X \to B)$.

Dependent types:
$\Pi(X : \mathcal{U}).(X \to A) \to (X \to B)$.

Suppose $B = \mathcal{U}$:
$\mathit{leak} : \Pi(X : \mathcal{U}).(X \to A) \to (X \to \mathcal{U})$
$\mathit{leak}\ X\ r\ x = X$.
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:
$\forall X.(X \to A) \to (X \to B)$.

Dependent types:
$\Pi(X : \mathcal{U}).(X \to A) \to (X \to B)$.

Suppose $B = \mathcal{U}$:
$\mathit{leak} : \Pi(X : \mathcal{U}).(X \to A) \to (X \to \mathcal{U})$
$\mathit{leak}\ X\ r\ x = X$.
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:
$\forall X.(X \to A) \to (X \to B)$.

Dependent types:
$\Pi(X : \mathcal{U}).(X \to A) \to (X \to B)$.

Suppose $B = \mathcal{U}$:
$leak : \Pi(X : \mathcal{U}).(X \to A) \to (X \to \mathcal{U})$
$leak\ X\ r\ x = X$.
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

Parametric quantifiers

Non-parametric functions

$$f : \Pi(x : A).B\,x, \qquad f : A \to B$$

can use argument **as data**.

$$\frac{A : \mathcal{U} \qquad B : A \to \mathcal{U}}{\Pi(x : A).B\,x : \mathcal{U}}$$

Parametric functions

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

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

$$\frac{A : \mathcal{U} \qquad B : A \to \mathcal{U}}{\forall(x : A).B\,x : \mathcal{U}}$$

Non-parametric pairs

$$p : \Sigma(x : A).B\,x, \qquad p : A \times B$$

Parametric pairs (packs)

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

*if all elements of A are related.

Adding parametric quantifiers

Non-parametric quantifiers

Non-parametric functions

$$f : \Pi(x : A).B\,x, \qquad f : A \to B$$

can use argument **as data**.

$$\frac{A : \mathcal{U} \qquad B : A \to \mathcal{U}}{\Pi(x : A).B\,x : \mathcal{U}}$$

Non-parametric pairs

$$p : \Sigma(x : A).B\,x, \qquad p : A \times B$$

Parametric quantifiers

Parametric functions

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

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

$$\frac{A : \mathcal{U} \qquad B : A \to \mathcal{U}}{\forall(x : A).B\,x : \mathcal{U}}$$

Parametric pairs (packs)

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

*if all elements of A are related.

Adding parametric quantifiers

Non-parametric quantifiers

Non-parametric functions

$$f : \Pi(x : A).B\,x, \qquad f : A \to B$$

can use argument **as data.**

$$\frac{A : \mathcal{U} \qquad B : A \to \mathcal{U}}{\Pi(x : A).B\,x : \mathcal{U}}$$

Non-parametric pairs

$$p : \Sigma(x : A).B\,x, \qquad p : A \times B$$

Parametric quantifiers

Parametric functions

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

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

$$\frac{A : \mathcal{U} \qquad B : A \to \mathcal{U}}{\forall(x : A).B\,x : \mathcal{U}}$$

Parametric pairs (packs)

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

*if all elements of A are related.

Adding parametric quantifiers

Non-parametric quantifiers

Non-parametric functions

$$f : \Pi(x : A).B\,x, \qquad f : A \to B$$

can use argument **as data**.

$$\frac{A : \mathcal{U} \qquad B : A \to \mathcal{U}}{\Pi(x : A).B\,x : \mathcal{U}}$$

Non-parametric pairs

$$p : \Sigma(x : A).B\,x, \qquad p : A \times B$$

Parametric quantifiers

Parametric functions

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

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

$$\frac{A : \mathcal{U} \qquad B : A \to \mathcal{U}}{\forall(x : A).B\,x : \mathcal{U}}$$

Parametric pairs (packs)

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

*if all elements of A are related.

$\forall$ is watching you

Use $\forall$ if you want representation independence:
$\forall(X : \mathcal{U}).(X \to A) \to (X \to B)$.

The following is now ill-typed:
$\mathit{leak} : \forall(X : \mathcal{U}).(X \to A) \to (X \to \mathcal{U})$
$\mathit{leak}\ X\ r\ x = X$,
since we cannot use X as data.

$\forall$ is watching you

Use $\forall$ if you want representation independence:
$\forall(X : \mathcal{U}).(X \to A) \to (X \to B)$.

The following is now ill-typed:
$\mathit{leak} : \forall(X : \mathcal{U}).(X \to A) \to (X \to \mathcal{U})$
$\mathit{leak}\ X\ r\ x = X$,
since we cannot use X as data.

Tools:

- Relational interval "type":
  $0 \frown 1 : \mathbb{I}$ *(cf. Bernardy, Coquand and Moulin (2015), Cohen, Coquand, Huber, Mörtberg (2016))*
- Graph type of $r : X \to A$
  $/r/ : \mathbb{I} \to \mathcal{U}$
  $/r/0 = X \quad \frown \quad /r/1 = A$
  (not a primitive)

$$
\begin{tikzcd}
0 : \mathbb{I} \arrow[dd, dotted, no head] & X \arrow[rr, "r"] \arrow[dd, dotted, no head] \arrow[dr, dotted, no head] & & A \arrow[dd, dotted, no head] \\
& & & \\
i : \mathbb{I} \arrow[dd, dotted, no head] & X \arrow[r, "push\ i"'] \arrow[dd, dotted, no head] & /r/i \arrow[r, "pull\ i"] \arrow[ddr, dotted, no head] & A \arrow[dd, dotted, no head] \\
& & & \\
1 : \mathbb{I} & X \arrow[rr, "r"'] & & A
\end{tikzcd}
$$

Lemma

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

Proof:

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

Semantically: $0 \frown 1 \Rightarrow p\ 0 = p\ 1$

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

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

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

An internal proof

Tools:

- Relational interval "type":
  $0 \frown 1 : \mathbb{I}$ *(cf. Bernardy, Coquand and Moulin (2015), Cohen, Coquand, Huber, Mörtberg (2016))*
- Graph type of $r : X \to A$
  $/r/ : \mathbb{I} \to \mathcal{U}$
  $/r/0 = X \quad \frown \quad /r/1 = A$
  (not a primitive)

$$\begin{tikzcd}
0 : \mathbb{I} & X \arrow[rr, "r"] & & A \\
i : \mathbb{I} & X \arrow[r, "\text{push } i"'] & /r/i \arrow[r, "\text{pull } i"] & A \\
1 : \mathbb{I} & X \arrow[rr, "r"'] & & A
\end{tikzcd}$$

Lemma

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

Proof:

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

Semantically: $0 \frown 1 \Rightarrow p\ 0 = p\ 1$

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

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

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

Tools:

- Relational interval "type":
  $0 \frown 1 : \mathbb{I}$ *(cf. Bernardy, Coquand and Moulin (2015), Cohen, Coquand, Huber, Mörtberg (2016))*
- Graph type of $r : X \to A$
  $/r/ : \mathbb{I} \to \mathcal{U}$
  $/r/0 = X \quad \frown \quad /r/1 = A$
  (not a primitive)

$$\begin{tikzcd}
0 : \mathbb{I} \arrow[d, dash, dotted] & X \arrow[rr, "r"] \arrow[d, dash, dotted] \arrow[dr, dash, dotted] & & A \arrow[d, dash, dotted] \\
i : \mathbb{I} \arrow[d, dash, dotted] & X \arrow[r, "\text{push } i"'] \arrow[d, dash, dotted] & /r/i \arrow[r, "\text{pull } i"] \arrow[dr, dash, dotted] & A \arrow[d, dash, dotted] \\
1 : \mathbb{I} & X \arrow[rr, "r"'] & & A
\end{tikzcd}$$

Lemma

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

Proof:

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

Semantically: $0 \frown 1 \Rightarrow p\ 0 = p\ 1$

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

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

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

An internal proof

Tools:

- Relational interval "type":
  $0 \frown 1 : \mathbb{I}$ *(cf. Bernardy, Coquand and Moulin (2015), Cohen, Coquand, Huber, Mörtberg (2016))*
- Graph type of $r : X \to A$
  $/r/ : \mathbb{I} \to \mathcal{U}$
  $/r/0 = X \quad \frown \quad /r/1 = A$
  (not a primitive)

$$
\begin{tikzcd}
0 : \mathbb{I} \arrow[d, dotted, no head] & X \arrow[rr, "r"] \arrow[d, dotted, no head] \arrow[dr, dotted, no head] & & A \arrow[d, dotted, no head] \\
i : \mathbb{I} \arrow[d, dotted, no head] & X \arrow[r, "\text{push } i"'] \arrow[d, dotted, no head] & /r/i \arrow[r, "\text{pull } i"] \arrow[dr, dotted, no head] & A \arrow[d, dotted, no head] \\
1 : \mathbb{I} & X \arrow[rr, "r"'] & & A
\end{tikzcd}
$$

Lemma

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

Proof:

We define: $p : \forall(\_ : \mathbb{I}).B$
Semantically: $0 \frown 1 \Rightarrow p\ 0 = p\ 1$
$p\ i = g\ (/r/i)\ (\text{pull } i)\ (\text{push } i\ x)$
$p\ 0 = g\ X\ r\ (\mathrm{id}\ x)$
$p\ 1 = g\ A\ \mathrm{id}\ (r\ x)$ $\square$

Tools:

- Relational interval “type”:
  $0 \frown 1 : \mathbb{I}$ *(cf. Bernardy, Coquand and Moulin (2015), Cohen, Coquand, Huber, Mörtberg (2016))*
- Graph type of $r : X \to A$
  $/r/ : \mathbb{I} \to \mathcal{U}$
  $/r/0 = X \quad \frown \quad /r/1 = A$
  (not a primitive)

$$
\begin{tikzcd}
0 : \mathbb{I} \arrow[d, dash, dotted] & X \arrow[rr, "r"] \arrow[d, dash, dotted] \arrow[dr, dash, dotted] & & A \arrow[d, dash, dotted] \\
i : \mathbb{I} \arrow[d, dash, dotted] & X \arrow[r, "\text{push } i"'] \arrow[d, dash, dotted] & /r/i \arrow[r, "\text{pull } i"] \arrow[dr, dash, dotted] & A \arrow[d, dash, dotted] \\
1 : \mathbb{I} & X \arrow[rr, "r"'] & & A
\end{tikzcd}
$$

Lemma

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

Proof:

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

Semantically: $0 \frown 1 \Rightarrow p\ 0 = p\ 1$

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

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

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

Tools:

- Relational interval "type":
  $0 \frown 1 : \mathbb{I}$ *(cf. Bernardy, Coquand and Moulin (2015), Cohen, Coquand, Huber, Mörtberg (2016))*
- Graph type of $r : X \to A$
  $/r/ : \mathbb{I} \to \mathcal{U}$
  $/r/0 = X \quad \frown \quad /r/1 = A$
  (not a primitive)

$$
\begin{tikzcd}
0 : \mathbb{I} \arrow[d, dash, dotted] & X \arrow[rr, "r"] \arrow[d, dash, dotted] \arrow[dr, dash, dotted] & & A \arrow[d, dash, dotted] \\
i : \mathbb{I} \arrow[d, dash, dotted] & X \arrow[r, "\text{push } i"'] \arrow[d, dash, dotted] & /r/i \arrow[r, "\text{pull } i"] \arrow[dr, dash, dotted] & A \arrow[d, dash, dotted] \\
1 : \mathbb{I} & X \arrow[rr, "r"'] & & A
\end{tikzcd}
$$

Lemma

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

Proof:

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

Semantically: $0 \frown 1 \Rightarrow p\ 0 = p\ 1$

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

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

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

An internal proof

Tools:

- Relational interval "type":
  $0 \frown 1 : \mathbb{I}$ *(cf. Bernardy, Coquand and Moulin (2015), Cohen, Coquand, Huber, Mörtberg (2016))*
- Graph type of $r : X \to A$
  $/r/ : \mathbb{I} \to \mathcal{U}$
  $/r/0 = X \quad \frown \quad /r/1 = A$
  (not a primitive)

$$
\begin{tikzcd}
0 : \mathbb{I} \arrow[d, dotted, no head] & X \arrow[rr, "r"] \arrow[d, dotted, no head] \arrow[dr, dotted, no head] & & A \arrow[d, dotted, no head] \\
i : \mathbb{I} \arrow[d, dotted, no head] & X \arrow[r, "\text{push } i"'] \arrow[d, dotted, no head] & /r/i \arrow[r, "\text{pull } i"] \arrow[dr, dotted, no head] & A \arrow[d, dotted, no head] \\
1 : \mathbb{I} & X \arrow[rr, "r"'] & & A
\end{tikzcd}
$$

Lemma

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

Proof:

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

Semantically: $0 \frown 1 \Rightarrow p\ 0 = p\ 1$

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

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

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

Tools:

- Relational interval "type":
  $0 \frown 1 : \mathbb{I}$ *(cf. Bernardy, Coquand and Moulin (2015), Cohen, Coquand, Huber, Mörtberg (2016))*
- Graph type of $r : X \to A$
  $/r/ : \mathbb{I} \to \mathcal{U}$
  $/r/0 = X \quad \frown \quad /r/1 = A$
  (not a primitive)

$$\begin{tikzcd}
0 : \mathbb{I} \arrow[d, dash, dotted] & X \arrow[rr, "r"] \arrow[d, dash, dotted] \arrow[dr, dash, dotted] & & A \arrow[d, dash, dotted] \\
i : \mathbb{I} \arrow[d, dash, dotted] & X \arrow[r, "push\ i"'] \arrow[d, dash, dotted] & /r/i \arrow[r, "pull\ i"] \arrow[dr, dash, dotted] & A \arrow[d, dash, dotted] \\
1 : \mathbb{I} & X \arrow[rr, "r"'] & & A
\end{tikzcd}$$

Lemma

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

Proof:

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

Semantically: $0 \frown 1 \Rightarrow p\ 0 = p\ 1$

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

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

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

Contributions

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 (yet?) 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.

Contributions

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 (yet?) 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.

Contributions

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 (yet?) 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.

Contributions

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 (yet?) 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.

Contributions

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 (yet?) 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.

Contributions

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 (yet?) 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!

Related talks:

Normalization by Evaluation for Sized Dependent Types
Abel, Vezzosi, Winterhalter – Up next

A Fibrational Framework for Substructural and Modal Logics
Licata, Shulman, Riley – 13h @ FSCD

Questions?

Church initial algebras

Assume level-preserving functor F.

$\mathsf{Mu}_\ell = \forall(X : \mathcal{U}_\ell).(FX \to X) \to X.$
$\mathsf{mkMu}_\ell : F\mathsf{Mu}_\ell \to \mathsf{Mu}_\ell.$

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

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

Theorem (Initiality of Mu up to $\downarrow$)

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

$$\begin{tikzcd}
\mathsf{Mu}_{\ell+1} \arrow[d, "\downarrow_\ell"'] \arrow[dr, "\mathsf{fold}_{\ell+1}\ B\ \mathsf{mk}B"] & \\
\mathsf{Mu}_\ell \arrow[r, "f"'] & B
\end{tikzcd}$$

Church initial algebras

Assume level-preserving functor F.

$\mathsf{Mu}_\ell = \forall(X : \mathcal{U}_\ell).(FX \to X) \to X.$
$\mathsf{mkMu}_\ell : F\mathsf{Mu}_\ell \to \mathsf{Mu}_\ell.$

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

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

Theorem (Initiality of Mu up to $\downarrow$)

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

$$\begin{tikzcd}
\mathsf{Mu}_{\ell+1} \arrow[d, "\downarrow_\ell"'] \arrow[dr, "\mathsf{fold}_{\ell+1}\ B\ \mathsf{mk}B"] & \\
\mathsf{Mu}_\ell \arrow[r, "f"'] & B
\end{tikzcd}$$

Church initial algebras

Assume level-preserving functor F.

$\mathrm{Mu}_\ell = \forall(X : \mathcal{U}_\ell).(FX \to X) \to X.$
$\mathrm{mkMu}_\ell : F\mathrm{Mu}_\ell \to \mathrm{Mu}_\ell.$

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

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

Theorem (Initiality of Mu up to $\downarrow$)

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

$$\begin{tikzcd}
\mathrm{Mu}_{\ell+1} \arrow[d, "\downarrow_\ell"'] \arrow[dr, "\mathrm{fold}_{\ell+1}\ B\ \mathrm{mk}B"] & \\
\mathrm{Mu}_\ell \arrow[r, "f"'] & B
\end{tikzcd}$$

Church initial algebras

Assume level-preserving functor F.

$\mathrm{Mu}_\ell = \forall(X : \mathcal{U}_\ell).(FX \to X) \to X$.
$\mathrm{mkMu}_\ell : F\mathrm{Mu}_\ell \to \mathrm{Mu}_\ell$.

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

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

Theorem (Initiality of Mu up to $\downarrow$)

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

$$\begin{tikzcd} \mathrm{Mu}_{\ell+1} \arrow[d, "\downarrow_\ell"'] \arrow[dr, "\mathrm{fold}_{\ell+1}\ B\ \mathrm{mk}B"] & \\ \mathrm{Mu}_\ell \arrow[r, "f"'] & B \end{tikzcd}$$

Initial algebras using Size

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

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

Parametricity: side bounds are hidden

Works for finitely branching container functors (even indexed): $F\ T = \Sigma(c : C).(B\ c \to T)$.
Also final co-algebras (e.g. streams).

Initial algebras using Size

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

- By well-founded induction on n : 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): $F\ T = \Sigma(c : C).(B\ c \to T)$.
Also final co-algebras (e.g. streams).

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

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

Parametricity: side bounds are hidden

Works for finitely branching container functors (even indexed): $F\ T = \Sigma(c : C).(B\ c \to T)$.
Also final co-algebras (e.g. streams).

Initial algebras using Size

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

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

Parametricity: side bounds are hidden

Works for finitely branching container functors (even indexed): $F\ T = \Sigma(c : C).(B\ c \to T)$.
Also final co-algebras (e.g. streams).

Initial algebras using Size

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

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

Parametricity: side bounds are hidden

Works for finitely branching container functors (even indexed):
$F\ T = \Sigma(c : C).(B\ c \to T)$.
Also final co-algebras (e.g. streams).

Initial algebras using Size

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

- By well-founded induction on n : 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.
- List $A := \exists n.\widehat{\text{List}}\ A\ n$.

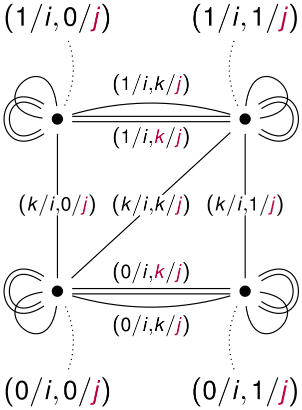
Parametricity: side bounds are hidden

Works for finitely branching container functors (even indexed):
$F\ T = \Sigma(c : C).(B\ c \to T)$.
Also final co-algebras (e.g. streams).

Example of a bridge/path cubical set

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

legend:



Related work

citation	source	target	journey	model	IEL proof
Reynolds, 1983	Sys F			set th.	**yes**
Abadi, Cardelli, Curien, 1993	Sys F	Sys $\mathscr{R}$	external		**yes**
Plotkin & Abadi, 1993	Sys F	Sys F + logic	external		**yes**
Wadler, 2007	Sys F	Sys F + logic	external		**yes**
Takeuti, 2001	$\mathscr{X} \in \lambda$-cube	$\mathscr{Y} \in \lambda$-cube	external		for small types
Bernardy, Jansson, Paterson, 2012	any PTS	other PTS	external		no
Krishnaswami & Dreyer, 2013	dependent types			Q-PER	only some corollaries
Atkey, Ghani, Johann, 2014	dependent types			presh.	for small types
Bernardy, Coquand, Moulin, 2015	dependent types + param. operators	same as source	**internal**	presh.	no
This work	dependent types + param. operators, $\forall$, $\exists$	same as source	**internal**	presh.	**yes**

Parametricity, Shape-irrelevance, irrelevance

4 functors on bridge/path cubical sets:

- id Non-parametricity (continuity)
- $\sharp$ Parametricity
- $\bullet\bullet$ Shape-irrelevance
- $\bullet$ Irrelevance

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

Abel et al. have: $[\![\text{Size}]\!] = [\![\mathbb{N}]\!]$.
We have: $[\![\text{Size}]\!] = \bullet\bullet [\![\mathbb{N}]\!]$. Hence, $\sharp [\![\text{Size}]\!] = \bullet [\![\mathbb{N}]\!]$.

ParamDTT	Abel	$[\![\text{domain}]\!]$
$\Pi(i : \text{Size}).A\ i$	$\bullet\bullet(i : \text{Size}) \to A\ i$	$\bullet\bullet [\![\mathbb{N}]\!]$
$\forall(i : \text{Size}).A\ i$	$\bullet(i : \text{Size}) \to A\ i$	$\bullet [\![\mathbb{N}]\!]$