Internal and Observational Parametricity for Cubical Agda

Antoine Van Muylder[1] Andreas Nuyts[1] Dominique Devriese[1]

[1]KU Leuven

HoTT/UF 24
4 Apr 2024

Question

Best way to provide *free theorems* to the proof assistant user?

A free theorem

Conjecture (normal DTT)

$$p : (X : \mathsf{Type}) \to X \to X \to X$$

then

$$p \equiv \lambda X\, x\, y.\, x \qquad \text{OR} \qquad p \equiv \lambda X\, x\, y.\, y$$

A free theorem

Conjecture (normal DTT)

$$p : (X : \mathsf{Type}) \to X \to X \to X$$

then

$$p \equiv \lambda X\, x\, y.\, x \qquad \text{OR} \qquad p \equiv \lambda X\, x\, y.\, y$$

Intuition

- p can not case on $X : \mathsf{Type}$
- i.e. p is "parametric"

Defining parametricity

[Reynolds 1983] gived us a definition

p is parametric

$\iff$

- p preserves *logical relations*

Defining parametricity

[Reynolds 1983] gived us a definition

p is parametric

$\iff$

- *p* preserves *logical relations*
- i.e. *p* preserves *structure-compatible relations*

Defining parametricity

[Reynolds 1983] gived us a definition

p is parametric
$\iff$

- p preserves *logical relations*
- i.e. p preserves *structure-compatible relations*

for $p : (X : \mathsf{Type}) \to X \to X \to X$ this unfolds to

Defining parametricity

[Reynolds 1983] gived us a definition

p is parametric
$\iff$

- p preserves *logical relations*
- i.e. p preserves *structure-compatible relations*

for $p : (X : \mathsf{Type}) \to X \to X \to X$ this unfolds to

- $(R : X_0 \to X_1 \to \mathsf{Type}) \to$

Defining parametricity

[Reynolds 1983] gived us a definition

p is parametric

$\iff$

- p preserves *logical relations*
- i.e. p preserves *structure-compatible relations*

for $p : (X : \text{Type}) \to X \to X \to X$ this unfolds to

- $(R : X_0 \to X_1 \to \text{Type}) \to$
- $x0, x1$ such that $R\, x_0\, x_1 \to y_0\, y_1$ such that $R\, y_0\, y_1 \to$

Defining parametricity

[Reynolds 1983] gived us a definition

p is parametric
$\iff$

- p preserves *logical relations*
- i.e. p preserves *structure-compatible relations*

for $p : (X : \mathsf{Type}) \to X \to X \to X$ this unfolds to

- $(R : X_0 \to X_1 \to \mathsf{Type}) \to$
- $x0, x1$ such that $R\, x_0\, x_1 \to y_0\, y_1$ such that $R\, y_0\, y_1 \to$
- $R(p\, X_0\, x_0\, y_0)(p\, X_1\, x_1\, y_1)$

Defining parametricity

[Reynolds 1983] gived us a definition

p is parametric
$\iff$

- p preserves *logical relations*
- i.e. p preserves *structure-compatible relations*

for $p : (X : \mathsf{Type}) \to X \to X \to X$ this unfolds to

- $(R : X_0 \to X_1 \to \mathsf{Type}) \to$
- $x0, x1$ such that $R\, x_0\, x_1 \to y_0\, y_1$ such that $R\, y_0\, y_1 \to$
- $R(p\, X_0\, x_0\, y_0)(p\, X_1\, x_1\, y_1)$
- fact: $\quad p$ param. $\implies$ $(p \equiv \lambda\, X\, x\, y\, .\, x \quad \text{OR} \quad p \equiv \lambda\, X\, x\, y\, .\, y)$

2 solutions

Q: Bring free theorems to proof assistants?

A1:
Parametricity
translations
(only alluded to)

A2: Internally parametric DTTs
note: interval-based

As we will see, A1 & A2 offer opposite tradeoffs

Internally parametric DTTs. What?

- Introduced in [Moulin 2016]

Internally parametric DTTs. What?

- Introduced in [Moulin 2016]
- Extension of standard DTT (new types, terms, equations)

$$\frac{A : \mathsf{Type} \qquad a_0, a_1 : A}{\mathsf{Bridge}_A\, a_0\, a_1 : \mathsf{Type}}$$

Bridges are synthetic log. relations $\quad \overline{M} : \mathsf{Bridge}_{\mathsf{Mon}}\, M_0\, M_1$

Paths are synthetic isomorphisms $\quad \overline{M} : M_0 \equiv_{\mathsf{Mon}} M_1$

Internally parametric DTTs. What?

- Introduced in [Moulin 2016]
- Extension of standard DTT (new types, terms, equations)

$$\frac{A : \mathsf{Type} \qquad a_0, a_1 : A}{\mathsf{Bridge}_A\, a_0\, a_1 : \mathsf{Type}}$$

Bridges are synthetic log. relations $\overline{M} : \mathsf{Bridge}_{\mathsf{Mon}}\, M_0\, M_1$
Paths are synthetic isomorphisms $\overline{M} : M_0 \equiv_{\mathsf{Mon}} M_1$

- In this setting, param. = functions preserve bridges

$$\vdash \mathsf{intparam} : (p : (a : A) \to B\, a) \to (\overline{a} : \mathsf{Bridge}_A\, a_0\, a_1) \to \mathsf{BridgeP}_{x.\, B(\overline{a}\, x)}\, (p\, a_0)\, (p\, a_1)$$

$$\vdash \mathsf{intparam}\; p\, \overline{a} := \lambda x.\, p\, (\overline{a}\, x)$$

Internally parametric DTTs. What?

- Introduced in [Moulin 2016]
- Extension of standard DTT (new types, terms, equations)

$$\frac{A : \mathsf{Type} \qquad a_0, a_1 : A}{\mathsf{Bridge}_A\, a_0\, a_1 : \mathsf{Type}}$$

Bridges are synthetic log. relations $\quad \overline{M} : \mathsf{Bridge}_{\mathsf{Mon}}\, M_0\, M_1$

Paths are synthetic isomorphisms $\quad \overline{M} : M_0 \equiv_{\mathsf{Mon}} M_1$

- In this setting, param. = functions preserve bridges

$$\vdash \mathsf{intparam} : (p : (a : A) \to B\, a) \to (\overline{a} : \mathsf{Bridge}_A\, a_0\, a_1) \to \mathsf{BridgeP}_{x.\, B(\overline{a}\, x)}\, (p\, a_0)\, (p\, a_1)$$

$$\vdash \mathsf{intparam}\, p\, \overline{a} := \lambda x.\, p\, (\overline{a}\, x)$$

- (and free theorems can be proved from intparam + other prims)

Internally parametric DTTs. Why?

$$\vdash \mathsf{intparam} : (p : (a : A) \to B\, a) \to (\overline{a} : \mathsf{Bridge}_A\, a_0\, a_1) \to \mathsf{BridgeP}_{x.\,B(\overline{a}\,x)}\,(p\, f_0)\,(p\, f_1)$$

$$\vdash \mathsf{intparam} := \lambda\, p\, \overline{a}.\ \lambda x.\ p\,(\overline{a}\, x)$$

- Quantification on p is *internal* to the type theory

Internally parametric DTTs. Why?

$$\vdash \text{intparam} : (p : (a : A) \to B\, a) \to (\overline{a} : \text{Bridge}_A\, a_0\, a_1) \to \text{BridgeP}_{x.\, B(\overline{a}\, x)}\, (p\, f_0)\, (p\, f_1)$$

$$\vdash \text{intparam} := \lambda\, p\, \overline{a}.\, \lambda x.\, p\, (\overline{a}\, x)$$

- Quantification on p is *internal* to the type theory
- This allows e.g. $\vdash \text{boolChurch} : (\forall X.\, X \to X \to X) \simeq \text{Bool}$

Internally parametric DTTs. Why?

$\vdash \mathsf{intparam} : (p : (a : A) \to B\, a) \to (\overline{a} : \mathsf{Bridge}_A\, a_0\, a_1) \to \mathsf{BridgeP}_{x.\, B(\overline{a}\, x)}\, (p\, f_0)\, (p\, f_1)$

$\vdash \mathsf{intparam} := \lambda\, p\, \overline{a}.\ \lambda x.\ p\, (\overline{a}\, x)$

- Quantification on p is *internal* to the type theory
- This allows e.g. $\vdash \mathsf{boolChurch} : (\forall X.\, X \to X \to X) \simeq \mathsf{Bool}$
- not achieved by param. translations

Internally parametric DTTs. Why?

$$\vdash \mathrm{intparam} : (p : (a : A) \to B\, a) \to (\overline{a} : \mathrm{Bridge}_A\, a_0\, a_1) \to \mathrm{BridgeP}_{x.\, B(\overline{a}\, x)}\, (p\, f_0)\, (p\, f_1)$$

$$\vdash \mathrm{intparam} := \lambda\, p\, \overline{a}.\, \lambda x.\, p\, (\overline{a}\, x)$$

- Quantification on p is *internal* to the type theory
- This allows e.g. $\vdash \mathrm{boolChurch} : (\forall X.\, X \to X \to X) \simeq \mathrm{Bool}$
- not achieved by param. translations

Problem: there are nice systems but no implementations

Internally parametric DTTs. Why?

$$\vdash \mathsf{intparam} : (p : (a : A) \to B\,a) \to (\overline{a} : \mathsf{Bridge}_A\, a_0\, a_1) \to \mathsf{BridgeP}_{x.\,B(\overline{a}\,x)}\,(p\, f_0)\,(p\, f_1)$$

$$\vdash \mathsf{intparam} := \lambda\, p\, \overline{a}.\, \lambda x.\, p\,(\overline{a}\, x)$$

- Quantification on p is *internal* to the type theory
- This allows e.g. $\vdash \mathsf{boolChurch} : (\forall X.\, X \to X \to X) \simeq \mathsf{Bool}$
- not achieved by param. translations

Problem: there are nice systems but no implementations

E.g. [Cavallo & Harper 2021] (CH):
Internal parametricity for cubical type theory extends HoTT!

Contrib. #1: Agda --bridges

Agda --bridges implements (variant of) CH on top of Agda --cubical

$$\begin{array}{rcl} \text{HoTT (cart. CTT)} & \longrightarrow & \text{CH} \\ \text{Agda --cubical} & \longrightarrow & \text{Agda --bridges} \end{array}$$

Contrib. #1: Agda --bridges

Agda --bridges implements (variant of) CH on top of Agda --cubical

$$\begin{array}{rcl} \text{HoTT (cart. CTT)} & \longrightarrow & \text{CH} \\ \text{Agda --cubical} & \longrightarrow & \text{Agda --bridges} \end{array}$$

- First practical proof assistant with internal parametricity

Contrib. #1: Agda --bridges

Agda --bridges implements (variant of) CH on top of Agda --cubical

$$\begin{array}{rcl} \text{HoTT (cart. CTT)} & \longrightarrow & \text{CH} \\ \text{Agda --cubical} & \longrightarrow & \text{Agda --bridges} \end{array}$$

- First practical proof assistant with internal parametricity
- Agda --bridges successfully typechecks the `cubical` library

Contrib. #1: Agda --bridges

Agda --bridges implements (variant of) CH on top of Agda --cubical

$$\begin{array}{rcl} \text{HoTT (cart. CTT)} & \longrightarrow & \text{CH} \\ \text{Agda --cubical} & \longrightarrow & \text{Agda --bridges} \end{array}$$

- First practical proof assistant with internal parametricity
- Agda --bridges successfully typechecks the `cubical` library
- Thus univalence and funExt available

Agda --bridges tour (1)

In Agda --bridges...

Agda --bridges tour (1)

In Agda --bridges...

- $\exists$ bridge interval BI and $\mathsf{bi0}, \mathsf{bi1} : \mathsf{BI}$

Agda --bridges tour (1)

In Agda --bridges...

- $\exists$ bridge interval BI and bi0, bi1 : BI
- $\exists$ bridge types $\text{Bridge}_A\ a_0\ a_1$ at each A : Type, $a_0, a_1 : A$

Agda --bridges tour (1)

In Agda --bridges...

- $\exists$ bridge interval BI and $\mathsf{bi0}, \mathsf{bi1} : \mathsf{BI}$
- $\exists$ bridge types $\mathsf{Bridge}_A\, a_0\, a_1$ at each $A : \mathsf{Type}$, $a_0, a_1 : A$
- Essentially

$$\mathsf{Bridge}_A\, a_0\, a_1 \approx \{\overline{a} : \mathsf{BI} \to A \mid \overline{a}\, \mathsf{bi0} = a_0,\ \overline{a}\, \mathsf{bi1} = a_1\}$$

Agda --bridges tour (1)

In Agda --bridges...

- $\exists$ bridge interval BI and bi0, bi1 : BI
- $\exists$ bridge types $\mathrm{Bridge}_A\, a_0\, a_1$ at each $A : \mathrm{Type},\ a_0, a_1 : A$
- Essentially

$$\mathrm{Bridge}_A\, a_0\, a_1 \approx \{\overline{a} : \mathrm{BI} \to A \mid \overline{a}\, \mathrm{bi0} = a_0,\ \overline{a}\, \mathrm{bi1} = a_1\}$$

Rules enforce: BI inputs are consumed sub-structurally ("affinely")
Some equations of CH can not be stated without that
Impl: raise freshness constraints at TC time
Extension of @tick annotation from Agda --guarded

Agda --bridges tour (1)

In Agda --bridges...

- $\exists$ bridge interval BI and bi0, bi1 : BI
- $\exists$ bridge types $\mathrm{Bridge}_A\, a_0\, a_1$ at each A : Type, $a_0, a_1 : A$
- Essentially

$$\mathrm{Bridge}_A\, a_0\, a_1 \approx \{\overline{a} : \mathrm{BI} \to A \mid \overline{a}\,\mathrm{bi0} = a_0,\ \overline{a}\,\mathrm{bi1} = a_1\}$$

  Rules enforce: BI inputs are consumed sub-structurally ("affinely")
  Some equations of CH can not be stated without that
  Impl: raise freshness constraints at TC time
  Extension of @tick annotation from Agda --guarded

- Similarly, $\exists$ types of dependent bridges BridgeP... $(\overline{a} : \forall(x : \mathrm{BI})...)$

Agda --bridges tour (2)

Other Agda --bridges primitives

$$\mathsf{extent}\, A\, B\, (f_0\, f_1 : A \to B) :$$
$$(\forall a_0\, a_1.\, \mathsf{Bridge}_A a_0\, a_1 \to \mathsf{Bridge}_B\, (f_0\, a_0)\, (f_1\, a_1)) \quad \longrightarrow \quad \mathsf{Bridge}_{A \to B}\, f_0\, f_1$$

$$\mathsf{Gel}\, A_0\, A_1 : (A_0 \to A_1 \to \mathsf{Type}) \quad \longrightarrow \quad \mathsf{Bridge}_{\mathsf{Type}}\, A_0\, A_1$$

$+$ rules proving they are $\simeq$

Also: redesigned transp, hcomp operations from --cubical   See paper

Low level free theorem in Agda --bridges

```
lowChurchBool : (∀ (X : Type) → X → X → X) ≃ Bool -- Church encoding

lowChurchBool = isoToEquiv (iso chToBool boolToCh (λ { true → refl ; false → refl })
  λ k → funExt λ A → funExt λ t → funExt λ f → param-prf k A t f)
  where
    boolToCh : Bool → (∀ (X : Type) → X → X → X)
    boolToCh true X xt xf = xt
    boolToCh false X xt xf = xf

    chToBool : (∀ (X : Type) → X → X → X) → Bool
    chToBool k = k Bool true false

    module CH-inverse-cond (k : ∀ (X : Type) → X → X → X) (A : Type) (t f : A) where
      R : Bool → A → Type
      R = λ b a → (boolToCh b A t f) ≡ a
      k-Gelx : (@tick x : BI) → Gel Bool A R x → Gel Bool A R x → Gel Bool A R x
      k-Gelx x = k (Gel Bool A R x)
      k-Gelx-gel-gel : (@tick x : BI) → Gel Bool A R x
      k-Gelx-gel-gel x = k-Gelx x (gel true t (refl) x) ((gel false f (refl) x))
      asBdg : BridgeP (λ x → Gel Bool A R x) (k Bool true false) (k A t f)
      asBdg x = k-Gelx-gel-gel x
      param-prf : R (k Bool true false) (k A t f)
      param-prf = ungel {R = R} λ x → asBdg x
    open CH-inverse-cond
```

Drawbacks

Low-level proofs work but

- Require familiarity with extent, Gel, substructural BI
- Lack compositionality. How to reuse for similar free theorems?
- From experience, do not scale

Internally param. DTT vs translations

	Low-level Agda --bridges	Param. translation
$\mathsf{Bool} \simeq \forall X.\, X \to X \to X$	✔	✘
proof work	✘ (understand Gel,...)	✔ (call the transl.)
proof reuse	✘	/

Internally param. DTT vs translations

	Low-level Agda --bridges	Param. translation
Bool $\simeq \forall X.\, X \to X \to X$	✔	✘
proof work	✘ (understand Gel,...)	✔ (call the transl.)
proof reuse	✘	/

Contrib. # 2: attempt at merging 2 methods
In 2 steps.

Step 1: Structure relatedness principle is enough

param = preservation of **logical relations** (not just bridges)
How to get param?
E.g. for $p : \forall X.\, X \to X \to X$

Step 1: Structure relatedness principle is enough

param = preservation of **logical relations** (not just bridges)
How to get param?
E.g. for $p : \forall X.\, X \to X \to X$

$$\begin{tikzcd}
\begin{array}{c}\forall x_0\, x_1\, (x_0\, R\, x_1). \\ \forall y_0\, y_1\, (y_0\, R\, y_1). \\ (p\, A_0\, x_0\, y_0)\, R\, (p\, A_1\, x_1\, y_1)\end{array} & & \mathsf{BridgeP}_{x.\overline{A}x \to \overline{A}x \to \overline{A}x}\, (p\, A_0)\, (p\, A_1) \arrow[ll, "\approx\mathsf{extent} \quad (\simeq)"'] \\
R : A_0 \to A_1 \to \mathsf{Type} \arrow[u, dashed, "\mathsf{param}\,(\mathrm{dep.})"] \arrow[rr, "\mathsf{Gel} \quad (\simeq)"] & & \overline{A} : \mathsf{Bridge}_{\mathsf{Type}}\, A_0\, A_1 \arrow[u, "\mathsf{intparam}\,(\mathrm{dep.})"']
\end{tikzcd}$$

Step 1: Structure relatedness principle is enough

param = preservation of **logical relations** (not just bridges)
How to get param?
E.g. for $p : \forall X.\, X \to X \to X$

$$
\begin{tikzcd}
\begin{array}{c} \forall x_0\, x_1\, (x_0 \mathrel{R} x_1). \\ \forall y_0\, y_1\, (y_0 \mathrel{R} y_1). \\ (p\, A_0\, x_0\, y_0) \mathrel{R} (p\, A_1\, x_1\, y_1) \end{array} & & \mathsf{BridgeP}_{x.\overline{A}x \to \overline{A}x \to \overline{A}x}\,(p\, A_0)\,(p\, A_1) \arrow[ll, "\approx\mathsf{extent} \quad (\simeq)"'] \\
R : A_0 \to A_1 \to \mathsf{Type} \arrow[u, dashed, "\mathsf{param}\,(\mathrm{dep.})"] \arrow[rr, "\mathsf{Gel} \quad (\simeq)"] & & \overline{A} : \mathsf{Bridge}_{\mathsf{Type}}\, A_0\, A_1 \arrow[u, "\mathsf{intparam}\,(\mathrm{dep.})"']
\end{tikzcd}
$$

SRP = horizontal $\simeq$'s at all types and type families

Stating the SRP (meta)

The SRP is a bridge version of the SIP.
The SRP at $A : \mathsf{Type}$ reads:

- There is an "observational" characterization
  $\mathsf{Bridge}_A\, a_0\, a_1 \;\simeq\; \ldots$

Stating the SRP (meta)

The SRP is a bridge version of the SIP.
The SRP at A : Type reads:

- There is an "observational" characterization
  $\mathrm{Bridge}_A\, a_0\, a_1 \;\simeq\; \ldots$

Examples

- For Type and Π, use Gel and extent
- For Mon, log. relations of monoids
- For hSet, hset-valued relations ,etc,...

Stating the SRP (meta)

The SRP is a bridge version of the SIP.
The SRP at A : Type reads:

- There is an "observational" characterization $\mathrm{Bridge}_A\, a_0\, a_1 \ \simeq \ \ldots$

Examples

- For Type and Π, use Gel and extent
- For Mon, log. relations of monoids
- For hSet, hset-valued relations ,etc,...

Assume A has SRP. The SRP at $B : A \to$ Type reads:

- There is an "observational" characterization $\mathrm{BridgeP}_{x.\,B(\overline{A}\,x)}\, b_0\, b_1 \ \simeq \ \ldots$

Example for family List : Type $\to$ Type
$\mathrm{BridgeP}_{x.\mathrm{List}\,(\overline{A}\,x)}\, as_0\, as_1 \ \simeq \{bs \mid \text{list of bridges}\}$

Proving the SRP

Proving the SRP by hand is difficult, more so than the SIP.

- extent has *two* bridge types in its domain c.f. funext
  $(\forall a_0\, a_1.\, \mathrm{Bridge}_A a_0\, a_1 \to \mathrm{Bridge}_B\, (f_0\, a_0)\, (f_1\, a_1)) \quad \simeq \quad \mathrm{Bridge}_{A \to B}\, f_0\, f_1$

Proving the SRP

Proving the SRP by hand is difficult, more so than the SIP.

- extent has *two* bridge types in its domain c.f. funext
  $(\forall a_0\, a_1.\, \mathrm{Bridge}_A a_0\, a_1 \to \mathrm{Bridge}_B\, (f_0\, a_0)\, (f_1\, a_1)) \quad \simeq \quad \mathrm{Bridge}_{A \to B}\, f_0\, f_1$
- No prop. J-rule for Bridge c.f. SIP at data types

Proving the SRP

Proving the SRP by hand is difficult, more so than the SIP.

- extent has *two* bridge types in its domain c.f. funext
  $(\forall a_0\, a_1.\, \mathsf{Bridge}_A a_0\, a_1 \to \mathsf{Bridge}_B\, (f_0\, a_0)\, (f_1\, a_1)) \quad \simeq \quad \mathsf{Bridge}_{A \to B}\, f_0\, f_1$
- No prop. J-rule for Bridge c.f. SIP at data types
- When proving the SIP, can discard proof-irrelevant fragment of type
  - If line of props $P : \mathsf{I} \to \mathsf{Type}$, $\mathsf{isContr}(\mathsf{PathP}_{i.P\,i}\, p_0\, p_1)$
  - If line of props $P : \mathsf{BI} \to \mathsf{Type}$, $\mathsf{isProp}(\mathsf{BridgeP}_{x.P\,x}\, p_0\, p_1)$

Proving the SRP

Proving the SRP by hand is difficult, more so than the SIP.

- extent has *two* bridge types in its domain c.f. funext
  $(\forall a_0\, a_1.\, \mathsf{Bridge}_A a_0\, a_1 \to \mathsf{Bridge}_B\,(f_0\, a_0)\,(f_1\, a_1)) \simeq \mathsf{Bridge}_{A\to B}\, f_0\, f_1$
- No prop. J-rule for Bridge c.f. SIP at data types
- When proving the SIP, can discard proof-irrelevant fragment of type
  - If line of props $P : \mathsf{I} \to \mathsf{Type}$, $\mathsf{isContr}(\mathsf{PathP}_{i.P\,i}\; p_0\; p_1)$
  - If line of props $P : \mathsf{BI} \to \mathsf{Type}$, $\mathsf{isProp}(\mathsf{BridgeP}_{x.P\,x}\; p_0\; p_1)$

Step 2: we have a DSL for SRP proofs. Idea:
{ types with the SRP } =: "relativistic refl. graphs"
form a model of DTT c.f. univalent groupoids

Step 2: a library for SRP proofs

The library is called ROTT (“relational observational type theory”)

features observational param as a “rule”

Step 2: a library for SRP proofs

The library is called ROTT ("relational observational type theory")
features observational param as a "rule"

```
-- 1) write type using ROTT library
X⊨X→X→X : DispRRG TypeRRG
X⊨X→X→X = →form (X⊨ElX) (→form X⊨ElX X⊨ElX) -- der. tree

-- 2) call param theorem
highChurchBool : (∀ (X : Type) → X → X → X) ≃ Bool -- Church encoding
highChurchBool = isoToEquiv (iso chToBool boolToCh (λ { true → refl ; false → refl })
  λ k → funExt λ A → funExt λ t → funExt λ f →
  param TypeRRG X⊨X→X→X k Bool A (λ b a → boolToCh b A t f ≡ a) -- param call
    true t refl false f refl)
  where
    boolToCh : Bool → (∀ (X : Type) → X → X → X)
    boolToCh true X xt xf = xt
    boolToCh false X xt xf = xf

    chToBool : (∀ (X : Type) → X → X → X) → Bool
    chToBool k = k Bool true false
```

Compared to low-level, proofs are

- compositional
- concise
- scalable

Compared to low-level, proofs are

- compositional
- concise
- scalable

Remarks

- ✘ no automation for derivation tree (yet?)

Compared to low-level, proofs are

- compositional
- concise
- scalable

Remarks

- ✘ no automation for derivation tree (yet?)
- ✘ no support for iterated parametricity

Compared to low-level, proofs are

- compositional
- concise
- scalable

Remarks

- ✘ no automation for derivation tree (yet?)
- ✘ no support for iterated parametricity
- ROTT is similar in scope to
  "*Internal parametricity, without an interval*" – Altenkirch et. al"
  Discussed in the present session. Note: ROTT has no syntax

Proved examples

- Simple free theorems involving List
- A scheme of Church encodings (containers)
- Reynolds abstraction thm. (pred.) Sys F $\longrightarrow$ RRG
- param for $k : \forall(M : \text{PreMnd}).\ \text{List}(M\,A) \to M\,A$ Voigtländer 09

wip

Making Agda --bridges compatible with HITs

Church encodings for QITs (e.g.) read as soundness-completeness

- $\mu\, F \;\simeq\; \forall (A : \mathrm{Alg}\, \mathrm{F}).\; A\,.\mathrm{carr}$
- "Operations obtainable syn. are operations that exist in all models"
- "equations in the syntax are equations in all models"...
- $\mathrm{FreeGrp}\, A \;\simeq\; \forall (Gg : (G : \mathrm{Grp}) \times (A \to G)).\; G.\mathrm{carr}$

What about coinductives?

- $\nu\, F \;\simeq\; \exists (C : \mathrm{Coalg}\, F).\; C.\mathrm{carr}$
- where $\exists\, A\, B \;=\; (\Sigma AB)/\mathrm{bridges}$