Universiteit Utrecht

[Faculty of **Science**
Information and Computing Sciences]

A Formal Comparison of Approaches to Datatype-Generic Programming

José Pedro Magalhães
joint work with Andres Löh

Utrecht University & Well-Typed LLP
`http://dreixel.net`

March 25, 2012

Outline

Introduction

Generic programming libraries, in Agda

- Regular
- PolyP
- Indexed functors
- Instant generics

Conversions

Conclusion

Setting

- There are many libraries for generic programming in Haskell
- Different approaches vary wildly in what datatypes they can encode (universe size) and in what functionality they can offer (expressiveness)
- There is a lot of duplicated code across different libraries
- Newcomers to the field never know what library to use
- Informal comparisons exist, but there are no embeddings, nor formalised statements

Setting

- There are many libraries for generic programming in Haskell
- Different approaches vary wildly in what datatypes they can encode (universe size) and in what functionality they can offer (expressiveness)
- There is a lot of duplicated code across different libraries
- Newcomers to the field never know what library to use
- Informal comparisons exist, but there are no embeddings, nor formalised statements

We intend to change this.

Generic programming libraries, in Agda

We have looked at five libraries:

- `regular`: simple library, one recursive position, no parameters

Generic programming libraries, in Agda

We have looked at five libraries:

- `regular`: simple library, one recursive position, no parameters
- `polyp`: historical approach, one recursive position, one parameter, and composition

Generic programming libraries, in Agda

We have looked at five libraries:

- `regular`: simple library, one recursive position, no parameters
- `polyp`: historical approach, one recursive position, one parameter, and composition
- `multirec`: multiple recursive positions, no parameters (omitted from this talk)

Generic programming libraries, in Agda

We have looked at five libraries:

- `regular`: simple library, one recursive position, no parameters
- `polyp`: historical approach, one recursive position, one parameter, and composition
- `multirec`: multiple recursive positions, no parameters (omitted from this talk)
- `indexed`: multiple recursive positions, multiple parameters, composition, and fixed points within the universe

Generic programming libraries, in Agda

We have looked at five libraries:

- `regular`: simple library, one recursive position, no parameters
- `polyp`: historical approach, one recursive position, one parameter, and composition
- `multirec`: multiple recursive positions, no parameters (omitted from this talk)
- `indexed`: multiple recursive positions, multiple parameters, composition, and fixed points within the universe
- `instant-generics`: coinductive approach with recursive codes

Outline

Regular—universe

```
module Regular where
  data Code : Set where
    U     :                  Code
    I     :                  Code
    _⊕_   : (F G : Code) → Code
    _⊗_   : (F G : Code) → Code
```

Regular—interpretation

$$
\begin{aligned}
[\![\_]\!] &: \mathsf{Code} \to (\mathsf{Set} \to \mathsf{Set}) \\
[\![\, \mathsf{U} \,]\!]\, \mathsf{A} &= \top \\
[\![\, \mathsf{I} \,]\!]\, \mathsf{A} &= \mathsf{A} \\
[\![\, \mathsf{F} \oplus \mathsf{G} \,]\!]\, \mathsf{A} &= [\![\, \mathsf{F} \,]\!]\, \mathsf{A} \uplus [\![\, \mathsf{G} \,]\!]\, \mathsf{A} \\
[\![\, \mathsf{F} \otimes \mathsf{G} \,]\!]\, \mathsf{A} &= [\![\, \mathsf{F} \,]\!]\, \mathsf{A} \times [\![\, \mathsf{G} \,]\!]\, \mathsf{A}
\end{aligned}
$$

$$
\begin{aligned}
&\textbf{data}\ \mu\ (\mathsf{F} : \mathsf{Code}) : \mathsf{Set}\ \textbf{where} \\
&\quad \langle\_\rangle : [\![\, \mathsf{F} \,]\!]\, (\mu\, \mathsf{F}) \to \mu\, \mathsf{F}
\end{aligned}
$$

Regular—map

$$
\begin{aligned}
&\mathsf{map} : \{A\ B : \mathsf{Set}\}\ (F : \mathsf{Code}) \\
&\quad \to (A \to B) \to [\![F]\!]\ A \to [\![F]\!]\ B \\
&\mathsf{map}\ \mathsf{U}\ \ f\ \_ = \mathsf{tt} \\
&\mathsf{map}\ \mathsf{I}\ \ f\ x = f\ x \\
&\mathsf{map}\ (F \oplus G)\ f\ (\mathsf{inj}_1\ x) = \mathsf{inj}_1\ (\mathsf{map}\ F\ f\ x) \\
&\mathsf{map}\ (F \oplus G)\ f\ (\mathsf{inj}_2\ x) = \mathsf{inj}_2\ (\mathsf{map}\ G\ f\ x) \\
&\mathsf{map}\ (F \otimes G)\ f\ (x , y) = \mathsf{map}\ F\ f\ x , \mathsf{map}\ G\ f\ y
\end{aligned}
$$

Outline

PolyP—universe

```
module PolyP where
  data Code : Set where
    U   :                   Code
    I   :                   Code
    P   :                   Code
    _⊕_ : (F G : Code) → Code
    _⊗_ : (F G : Code) → Code
    _⊚_ : (F G : Code) → Code
```

PolyP—interpretation

mutual

$$
\begin{array}{l}
[\![\_]\!] : \mathsf{Code} \to (\mathsf{Set} \to \mathsf{Set} \to \mathsf{Set}) \\
[\![\, \mathsf{U} \,]\!]\ \mathsf{A}\ \mathsf{R} = \top \\
[\![\, \mathsf{I} \,]\!]\ \mathsf{A}\ \mathsf{R} = \mathsf{R} \\
[\![\, \mathsf{P} \,]\!]\ \mathsf{A}\ \mathsf{R} = \mathsf{A} \\
[\![\, \mathsf{F} \oplus \mathsf{G} \,]\!]\ \mathsf{A}\ \mathsf{R} = [\![\, \mathsf{F} \,]\!]\ \mathsf{A}\ \mathsf{R} \uplus [\![\, \mathsf{G} \,]\!]\ \mathsf{A}\ \mathsf{R} \\
[\![\, \mathsf{F} \otimes \mathsf{G} \,]\!]\ \mathsf{A}\ \mathsf{R} = [\![\, \mathsf{F} \,]\!]\ \mathsf{A}\ \mathsf{R} \times [\![\, \mathsf{G} \,]\!]\ \mathsf{A}\ \mathsf{R} \\
[\![\, \mathsf{F} \circledcirc \mathsf{G} \,]\!]\ \mathsf{A}\ \mathsf{R} = \mu\ \mathsf{F}\ ([\![\, \mathsf{G} \,]\!]\ \mathsf{A}\ \mathsf{R})
\end{array}
$$

$$
\begin{array}{l}
\mathbf{data}\ \mu\ (\mathsf{F} : \mathsf{Code})\ (\mathsf{A} : \mathsf{Set}) : \mathsf{Set}\ \mathbf{where} \\
\quad \langle\_\rangle : [\![\, \mathsf{F} \,]\!]\ \mathsf{A}\ (\mu\ \mathsf{F}\ \mathsf{A}) \to \mu\ \mathsf{F}\ \mathsf{A}
\end{array}
$$

PolyP—map

$$
\begin{array}{l}
\textbf{mutual} \\
\quad \mathsf{map} : \{\mathsf{A\ B\ R\ S} : \mathsf{Set}\}\ (\mathsf{F} : \mathsf{Code}) \\
\qquad \to (\mathsf{A} \to \mathsf{B}) \to (\mathsf{R} \to \mathsf{S}) \to [\![\, \mathsf{F} \,]\!]\ \mathsf{A\ R} \to [\![\, \mathsf{F} \,]\!]\ \mathsf{B\ S} \\
\quad \mathsf{map\ U\ f\ g\ \_} = \mathsf{tt} \\
\quad \mathsf{map\ I\ f\ g\ x} = \mathsf{g\ x} \\
\quad \mathsf{map\ P\ f\ g\ x} = \mathsf{f\ x} \\
\quad \mathsf{map\ (F \oplus G)\ f\ g\ (inj_1\ x)} = \mathsf{inj_1\ (map\ F\ f\ g\ x)} \\
\quad \mathsf{map\ (F \oplus G)\ f\ g\ (inj_2\ x)} = \mathsf{inj_2\ (map\ G\ f\ g\ x)} \\
\quad \mathsf{map\ (F \otimes G)\ f\ g\ (x , y)} = \mathsf{map\ F\ f\ g\ x , map\ G\ f\ g\ y} \\
\quad \mathsf{map\ (F \circledcirc G)\ f\ g\ \langle\, x \,\rangle} = \langle\, \mathsf{map\ F\ (map\ G\ f\ g)} \\
\qquad\qquad \mathsf{(map\ (F \circledcirc G)\ f\ g)\ x} \,\rangle \\
\quad \mathsf{pmap} : \{\mathsf{A\ B} : \mathsf{Set}\}\ (\mathsf{F} : \mathsf{Code}) \\
\qquad \to (\mathsf{A} \to \mathsf{B}) \to \mu\ \mathsf{F\ A} \to \mu\ \mathsf{F\ B} \\
\quad \mathsf{pmap\ F\ f}\ \langle\, \mathsf{x} \,\rangle = \langle\, \mathsf{map\ F\ f\ (pmap\ F\ f)\ x} \,\rangle
\end{array}
$$

Outline

Introduction

Generic programming libraries, in Agda

Regular

PolyP

Indexed functors

Instant generics

Conversions

Conclusion

Indexed functors—universe

```
module Indexed where
  data Code (I : Set) (O : Set) : Set where
    U     :                                Code I O
    I     : I                            → Code I O
    !     : O                            → Code I O
    _⊕_   : (F G : Code I O)             → Code I O
    _⊗_   : (F G : Code I O)             → Code I O
    _⊚_   : {M : Set} → (F : Code M O)
          → (G : Code I M)               → Code I O
    Fix   : (F : Code (I ⊎ O) O)         → Code I O
```

Indexed functors—interpretation

```
Indexed   : Set → Set
Indexed I  =  I → Set

mutual
  ⟦_⟧ : {I O : Set} → Code I O → Indexed I → Indexed O
  ⟦ U       ⟧ r i = ⊤
  ⟦ I j     ⟧ r i = r j
  ⟦ ! j     ⟧ r i = i ≡ j
  ⟦ F ⊕ G ⟧ r i = ⟦ F ⟧ r i ⊎ ⟦ G ⟧ r i
  ⟦ F ⊗ G ⟧ r i = ⟦ F ⟧ r i × ⟦ G ⟧ r i
  ⟦ F ⊚ G ⟧ r i = ⟦ F ⟧ (⟦ G ⟧ r) i
  ⟦ Fix F   ⟧ r i = μ F r i

  data μ {I O : Set} (F : Code (I ⊎ O) O)
    (r : Indexed I) (o : O) : Set where
      ⟨_⟩ : ⟦ F ⟧ [r , μ F r] o → μ F r o
```

Indexed functors—map

$$
\begin{aligned}
&\mathsf{map} : \{I\ O : \mathsf{Set}\}\ \{R\ S : \mathsf{Indexed}\ I\}\ (F : \mathsf{Code}\ I\ O) \\
&\qquad \to R \rightrightarrows S \to [\![F]\!]\ R \rightrightarrows [\![F]\!]\ S \\
&\mathsf{map}\ U\ \ f\ i\ \_ = tt \\
&\mathsf{map}\ (I\ j)\ \ f\ i\ x = f\ j\ x \\
&\mathsf{map}\ (!\ j)\ \ f\ i\ x = x \\
&\mathsf{map}\ (F \oplus G)\ f\ i\ (inj_1\ x) = inj_1\ (\mathsf{map}\ F\ f\ i\ x) \\
&\mathsf{map}\ (F \oplus G)\ f\ i\ (inj_2\ x) = inj_2\ (\mathsf{map}\ G\ f\ i\ x) \\
&\mathsf{map}\ (F \otimes G)\ f\ i\ (x , y) = \mathsf{map}\ F\ f\ i\ x , \mathsf{map}\ G\ f\ i\ y \\
&\mathsf{map}\ (F \circledcirc G)\ f\ i\ x = \mathsf{map}\ F\ (\mathsf{map}\ G\ f)\ i\ x \\
&\mathsf{map}\ (\mathsf{Fix}\ F)\ \ f\ i\ \langle\ x\ \rangle = \langle\ \mathsf{map}\ F\ (f \parallel \mathsf{map}\ (\mathsf{Fix}\ F)\ f)\ i\ x\ \rangle
\end{aligned}
$$

Indexed functors—map

```
_⇉_ : {I : Set} → Indexed I → Indexed I → Set
R ⇉ S = (i : _) → R i → S i

_∥_ : {I J : Set} {A C : Indexed I} {B D : Indexed J}
      → A ⇉ C → B ⇉ D → [A , B] ⇉ [C , D]

map : {I O : Set} {R S : Indexed I} (F : Code I O)
      → R ⇉ S → ⟦ F ⟧ R ⇉ ⟦ F ⟧ S
map U         f i _          = tt
map (I j)     f i x          = f j x
map (! j)     f i x          = x
map (F ⊕ G)   f i (inj₁ x)   = inj₁ (map F f i x)
map (F ⊕ G)   f i (inj₂ x)   = inj₂ (map G f i x)
map (F ⊗ G)   f i (x , y)    = map F f i x , map G f i y
map (F ⊚ G)   f i x          = map F (map G f) i x
map (Fix F)   f i ⟨ x ⟩      = ⟨ map F (f ∥ map (Fix F) f) i x ⟩
```

Outline

Instant generics—universe

```
module InstantGenerics where
  mutual
    data Code : Set where
      U    :                        Code
      K    : Set                  → Code
      R    : ∞ Code               → Code
      _⊕_  : Code → Code          → Code
      _⊗_  : Code → Code          → Code
```

Instant generics—interpretation

$$
\begin{array}{llllll}
\textbf{data}\ [\![\_]\!] : \text{Code} \to \text{Set}\ \textbf{where} \\
\quad \text{tt} & : & & & & [\![\ \text{U}\ ]\!] \\
\quad \text{k} & : & \{\text{A} : \text{Set}\} & \to \text{A} & & \to [\![\ \text{K A}\ ]\!] \\
\quad \text{rec} & : & \{\text{C} : \infty\ \text{Code}\} & \to [\![\ \flat\ \text{C}\ ]\!] & & \to [\![\ \text{R C}\ ]\!] \\
\quad \text{inl} & : & \{\text{C D} : \text{Code}\} & \to [\![\ \text{C}\ ]\!] & & \to [\![\ \text{C} \oplus \text{D}\ ]\!] \\
\quad \text{inr} & : & \{\text{C D} : \text{Code}\} & \to [\![\ \text{D}\ ]\!] & & \to [\![\ \text{C} \oplus \text{D}\ ]\!] \\
\quad \_\,,\_ & : & \{\text{C D} : \text{Code}\} & \to [\![\ \text{C}\ ]\!] & \to [\![\ \text{D}\ ]\!] & \to [\![\ \text{C} \otimes \text{D}\ ]\!]
\end{array}
$$

Instant generics—sample generic function

Coinductive codes do not naturally define functors. We show an example generic function:

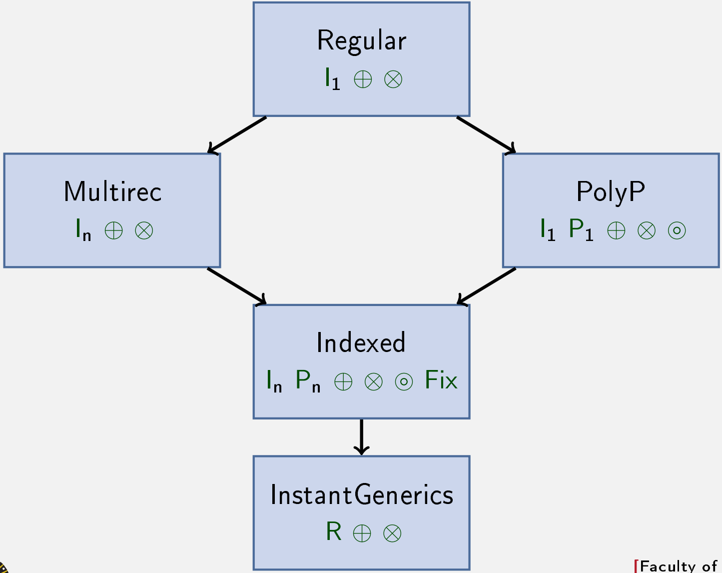
$$
\begin{aligned}
&\mathsf{size} : (A : \mathsf{Code}) \to [\![A]\!] \to \mathbb{N} \\
&\mathsf{size}\ \mathsf{U}\ x = 1 \\
&\mathsf{size}\ (\mathsf{K}\ A)\ (\mathsf{k}\ x) = 1 \\
&\mathsf{size}\ (\mathsf{R}\ C)\ (\mathsf{rec}\ x) = 1 + \mathsf{size}\ (\flat\ C)\ x \\
&\mathsf{size}\ (A \oplus B)\ (\mathsf{inl}\ x) = \mathsf{size}\ A\ x \\
&\mathsf{size}\ (A \oplus B)\ (\mathsf{inr}\ x) = \mathsf{size}\ B\ x \\
&\mathsf{size}\ (A \otimes B)\ (x , y) = \mathsf{size}\ A\ x + \mathsf{size}\ B\ y
\end{aligned}
$$

Encoding datatypes

```
module Example where
  data List (A : Set) : Set where
    nil  :                List A
    cons : A → List A → List A
  ListCp : Codep
  ListCp = Up ⊕p Pp ⊗p Ip
  ListCi : Codei ⊤ ⊤
  ListCi = Fixi (Ui ⊕i (Ii (inj1 tt)) ⊗i (Ii (inj2 tt)))
  ListCig : Codeig → Codeig
  ListCig A = Uig ⊕ig (A ⊗ig (Rig (♯ (ListCig A))))
```

Outline

Conversions



Regular to PolyP I

Code conversion:

$$
\begin{aligned}
{}_{r}{\Uparrow}^{p} &: \mathsf{Code}_r \to \mathsf{Code}_p \\
{}_{r}{\Uparrow}^{p}\ \mathsf{U}_r &= \mathsf{U}_p \\
{}_{r}{\Uparrow}^{p}\ \mathsf{I}_r &= \mathsf{I}_p \\
{}_{r}{\Uparrow}^{p}\ (F \oplus_r G) &= ({}_{r}{\Uparrow}^{p}\ F) \oplus_p ({}_{r}{\Uparrow}^{p}\ G) \\
{}_{r}{\Uparrow}^{p}\ (F \otimes_r G) &= ({}_{r}{\Uparrow}^{p}\ F) \otimes_p ({}_{r}{\Uparrow}^{p}\ G)
\end{aligned}
$$

Regular to PolyP II

Value conversion (one direction):

$$\mathsf{from_r} \;:\; \{\mathsf{A\ R} \;:\; \mathsf{Set}\}\ (\mathsf{C} \;:\; \mathsf{Code_r}) \to [\![\mathsf{C}]\!]_\mathsf{r}\ \mathsf{R} \to [\![{}_\mathsf{r}\!\Uparrow^\mathsf{p}\ \mathsf{C}]\!]_\mathsf{p}\ \mathsf{A\ R}$$

$$\begin{aligned}
\mathsf{from_r}\ \mathsf{U_r} &= \mathsf{id} \\
\mathsf{from_r}\ \mathsf{I_r} &= \mathsf{id} \\
\mathsf{from_r}\ (\mathsf{F} \oplus_\mathsf{r} \mathsf{G}) &= [\mathsf{inj}_1 \circ \mathsf{from_r}\ \mathsf{F}\,,\mathsf{inj}_2 \circ \mathsf{from_r}\ \mathsf{G}] \\
\mathsf{from_r}\ (\mathsf{F} \otimes_\mathsf{r} \mathsf{G}) &= <\ \mathsf{from_r}\ \mathsf{F} \circ \mathsf{proj}_1\,,\mathsf{from_r}\ \mathsf{G} \circ \mathsf{proj}_2\ >
\end{aligned}$$

$$\mathsf{from}\mu_\mathsf{r} \;:\; \{\mathsf{A} \;:\; \mathsf{Set}\}\ (\mathsf{C} \;:\; \mathsf{Code_r}) \to \mu_\mathsf{r}\ \mathsf{C} \to \mu_\mathsf{p}\ ({}_\mathsf{r}\!\Uparrow^\mathsf{p}\ \mathsf{C})\ \mathsf{A}$$

$$\mathsf{from}\mu_\mathsf{r}\ \mathsf{C}\ \langle \mathsf{x} \rangle_\mathsf{r} \;=\; \langle \mathsf{from_r}\ \mathsf{C}\ (\mathsf{map_r}\ \mathsf{C}\ (\mathsf{from}\mu_\mathsf{r}\ \mathsf{C})\ \mathsf{x}) \rangle_\mathsf{p}$$

Regular to PolyP III

Isomorphism proof (one direction):

$$
\begin{aligned}
&\mathsf{iso}_1 : \{A\ R : \mathsf{Set}\}\ (C : \mathsf{Code_r})\ \{x : [\![C]\!]_r\ R\} \\
&\quad \rightarrow \mathsf{to_r}\ \{A\}\ C\ (\mathsf{from_r}\ C\ x) \equiv x \\
&\mathsf{iso}_1\ \mathsf{U_r} = \mathsf{refl} \\
&\mathsf{iso}_1\ \mathsf{I_r} = \mathsf{refl} \\
&\mathsf{iso}_1\ (F \oplus_r G)\ \{\mathsf{inj}_1\ \_\} = \mathsf{cong}\ \mathsf{inj}_1\ (\mathsf{iso}_1\ F) \\
&\mathsf{iso}_1\ (F \oplus_r G)\ \{\mathsf{inj}_2\ \_\} = \mathsf{cong}\ \mathsf{inj}_2\ (\mathsf{iso}_1\ G) \\
&\mathsf{iso}_1\ (F \otimes_r G)\ \{\_ , \_\} = \mathsf{cong}_2\ \_ , \_\ (\mathsf{iso}_1\ F)\ (\mathsf{iso}_1\ G)
\end{aligned}
$$

Regular to PolyP IV

$$
\begin{aligned}
&\mathsf{iso}\mu_1 \;:\; \{\mathsf{A} : \mathsf{Set}\}\ (\mathsf{C} : \mathsf{Code_r})\ (\mathsf{x} : \mu_\mathsf{r}\ \mathsf{C}) \\
&\qquad \rightarrow \mathsf{to}\mu_\mathsf{r}\ \{\mathsf{A}\}\ \mathsf{C}\ (\mathsf{from}\mu_\mathsf{r}\ \mathsf{C}\ \mathsf{x}) \equiv \mathsf{x} \\
&\mathsf{iso}\mu_1\ \{\mathsf{A}\}\ \mathsf{C}\ \langle \mathsf{x} \rangle_\mathsf{r} \;=\; \mathsf{cong}\ \langle \_ \rangle_\mathsf{r}\ \$\ \mathsf{begin} \\
&\qquad \underline{\mathsf{to_r}}\ \{\mathsf{A}\}\ \mathsf{C}\ (\underline{\mathsf{map_p}}\ ({}_\mathsf{r}\Uparrow^\mathsf{p}\ \mathsf{C})\ \mathsf{id}\ (\mathsf{to}\mu_\mathsf{r}\ \mathsf{C}) \\
&\qquad\quad (\mathsf{from_r}\ \mathsf{C}\ (\mathsf{map_r}\ \mathsf{C}\ (\mathsf{from}\mu_\mathsf{r}\ \mathsf{C})\ \mathsf{x})))
\end{aligned}
$$

Regular to PolyP IV

$$\mathsf{iso}\mu_1 \;:\; \{A : \mathsf{Set}\}\ (C : \mathsf{Code_r})\ (x : \mu_r\ C)$$
$$\quad \rightarrow \mathsf{to}\mu_r\ \{A\}\ C\ (\mathsf{from}\mu_r\ C\ x) \equiv x$$
$$\mathsf{iso}\mu_1\ \{A\}\ C\ \langle x \rangle_r \;=\; \mathsf{cong}\ \langle \_ \rangle_r\ \$\ \mathsf{begin}$$
$$\quad \underline{\mathsf{to_r}}\ \{A\}\ C\ (\underline{\mathsf{map_p}\ (_r\Uparrow^p}\ C)\ \mathsf{id}\ (\mathsf{to}\mu_r\ C)$$
$$\qquad (\mathsf{from_r}\ C\ (\mathsf{map_r}\ C\ (\mathsf{from}\mu_r\ C)\ x)))$$
$$\equiv\langle\ \mathsf{mc}\ \{A\}\ C\ \rangle$$
$$\quad \mathsf{map_r}\ C\ (\mathsf{to}\mu_r\ \{A\}\ C)$$
$$\qquad (\mathsf{to_r}\ \{A\}\ C\ (\mathsf{from_r}\ C\ (\mathsf{map_r}\ C\ (\mathsf{from}\mu_r\ C)\ x)))$$

Regular to PolyP IV

```
isoμ1 : {A : Set} (C : Code_r) (x : μ_r C)
      → toμ_r {A} C (fromμ_r C x) ≡ x
isoμ1 {A} C ⟨x⟩_r = cong ⟨_⟩_r $ begin
    to_r {A} C (map_p (_r⇑^p C) id (toμ_r C)
      (from_r C (map_r C (fromμ_r C) x)))
  ≡⟨ mc {A} C ⟩
    map_r C (toμ_r {A} C)
      (to_r {A} C (from_r C (map_r C (fromμ_r C) x)))

mc : {A R1 R2 : Set} {x : _} {f : R1 → R2} (C : Code_r)
  → to_r {A} {R2} C (map_p (_r⇑^p C) id f x) ≡ map_r C f (to_r C x)
```

Regular to PolyP V

$$\mathsf{map_r}\ C\ (\mathsf{to}\mu_\mathsf{r}\ \{A\}\ C)$$
$$\quad (\underline{\mathsf{to_r}}\ \{A\}\ C\ (\underline{\mathsf{from_r}}\ C\ (\mathsf{map_r}\ C\ (\mathsf{from}\mu_\mathsf{r}\ C)\ x)))$$

Regular to PolyP V

$$
\begin{aligned}
& \mathsf{map_r}\ C\ (\mathsf{to}\mu_r\ \{A\}\ C) \\
& \quad (\underline{\mathsf{to_r}}\ \{A\}\ C\ (\underline{\mathsf{from_r}}\ C\ (\mathsf{map_r}\ C\ (\mathsf{from}\mu_r\ C)\ x))) \\
\equiv & \langle\ \mathsf{cong}\ (\mathsf{map_r}\ C\ (\mathsf{to}\mu_r\ \{A\}\ C))\ (\mathsf{iso_1}\ C)\ \rangle \\
& \underline{\mathsf{map_r}}\ C\ (\mathsf{to}\mu_r\ \{A\}\ C)\ (\underline{\mathsf{map_r}}\ C\ (\mathsf{from}\mu_r\ \{A\}\ C)\ x)
\end{aligned}
$$

Regular to PolyP V

$$
\begin{aligned}
& \mathsf{map_r}\ C\ (\mathsf{to}\mu_r\ \{A\}\ C) \\
& \quad (\underline{\mathsf{to_r}}\ \{A\}\ C\ (\underline{\mathsf{from_r}}\ C\ (\mathsf{map_r}\ C\ (\mathsf{from}\mu_r\ C)\ x))) \\
\equiv & \langle\ \mathsf{cong}\ (\mathsf{map_r}\ C\ (\mathsf{to}\mu_r\ \{A\}\ C))\ (\mathsf{iso}_1\ C)\ \rangle \\
& \underline{\mathsf{map_r}}\ C\ (\mathsf{to}\mu_r\ \{A\}\ C)\ (\underline{\mathsf{map_r}}\ C\ (\mathsf{from}\mu_r\ \{A\}\ C)\ x) \\
\equiv & \langle\ \mathsf{map}^{\circ}_{r}\ C\ \rangle \\
& \mathsf{map_r}\ C\ (\underline{\mathsf{to}\mu_r}\ C \circ \underline{\mathsf{from}\mu_r}\ C)\ x
\end{aligned}
$$

Regular to PolyP V

$$\mathsf{map_r}\ C\ (\mathsf{to}\mu_\mathsf{r}\ \{A\}\ C)$$
$$(\underline{\mathsf{to_r}}\ \{A\}\ C\ (\underline{\mathsf{from_r}}\ C\ (\mathsf{map_r}\ C\ (\mathsf{from}\mu_\mathsf{r}\ C)\ x)))$$
$$\equiv\langle\ \mathsf{cong}\ (\mathsf{map_r}\ C\ (\mathsf{to}\mu_\mathsf{r}\ \{A\}\ C))\ (\mathsf{iso_1}\ C)\ \rangle$$
$$\underline{\mathsf{map_r}}\ C\ (\mathsf{to}\mu_\mathsf{r}\ \{A\}\ C)\ (\underline{\mathsf{map_r}}\ C\ (\mathsf{from}\mu_\mathsf{r}\ \{A\}\ C)\ x)$$
$$\equiv\langle\ \mathsf{map_r^\circ}\ C\ \rangle$$
$$\mathsf{map_r}\ C\ (\underline{\mathsf{to}\mu_\mathsf{r}}\ C \circ \underline{\mathsf{from}\mu_\mathsf{r}}\ C)\ x$$
$$\equiv\langle\ \mathsf{map_r^\forall}\ C\ (\mathsf{to}\mu_\mathsf{r}\ C \circ \mathsf{from}\mu_\mathsf{r}\ C)\ \mathsf{id}\ (\mathsf{iso}\mu_1\ C)\ x\ \rangle$$
$$\mathsf{map_r}\ C\ \mathsf{id}\ x$$
$$\equiv\langle\ \mathsf{map_r^{id}}\ C\ \rangle$$
$$x \quad \square$$

Universiteit Utrecht

[Faculty of Science
Information and Computing Sciences]

PolyP to InstantGenerics

$$
\begin{aligned}
&{}_{p}\Uparrow^{ig} : \mathsf{Code}_p \to \mathsf{Set} \to \mathsf{Code}_{ig}\\
&{}_{p}\Uparrow^{ig}\ C\ A = {}_{p}\Uparrow^{ig\cdot}\ C\ C\ A\ \textbf{where}
\end{aligned}
$$

$$
\begin{aligned}
&{}_{p}\Uparrow^{ig\cdot} : \mathsf{Code}_p \to \mathsf{Code}_p \to \mathsf{Set} \to \mathsf{Code}_{ig}\\
&{}_{p}\Uparrow^{ig\cdot}\ C\ U_p\ A = U_{ig}\\
&{}_{p}\Uparrow^{ig\cdot}\ C\ I_p\ A = R_{ig}\ (\sharp\ {}_{p}\Uparrow^{ig\cdot}\ C\ C\ A)\\
&{}_{p}\Uparrow^{ig\cdot}\ C\ P_p\ A = K_{ig}\ A\\
&{}_{p}\Uparrow^{ig\cdot}\ C\ (F \oplus_p G)\ A = ({}_{p}\Uparrow^{ig\cdot}\ C\ F\ A) \oplus_{ig} ({}_{p}\Uparrow^{ig\cdot}\ C\ G\ A)\\
&{}_{p}\Uparrow^{ig\cdot}\ C\ (F \otimes_p G)\ A = ({}_{p}\Uparrow^{ig\cdot}\ C\ F\ A) \otimes_{ig} ({}_{p}\Uparrow^{ig\cdot}\ C\ G\ A)\\
&{}_{p}\Uparrow^{ig\cdot}\ C\ (F \circledcirc_p G)\ A = R_{ig}\ (\sharp\ {}_{p}\Uparrow^{ig\cdot}\ F\ F\ [\![({}_{p}\Uparrow^{ig\cdot}\ C\ G\ A)]\!]_{ig})
\end{aligned}
$$

Universiteit Utrecht

[Faculty of Science
Information and Computing Sciences]

Indexed to InstantGenerics

$$
\begin{aligned}
&\text{¡}\Uparrow^{ig} \;:\; \{I\; O \;:\; \mathsf{Set}\} \\
&\quad \rightarrow \mathsf{Code}_i\; I\; O \rightarrow (I \rightarrow \mathsf{Set}) \rightarrow (O \rightarrow \mathsf{Code}_{ig}) \\
&\text{¡}\Uparrow^{ig}\; C\; r\; o \;=\; \text{¡}\Uparrow^{ig\cdot}\; C\; (K_{ig} \circ r)\; o \;\textbf{where}
\end{aligned}
$$

$$
\begin{aligned}
&\text{¡}\Uparrow^{ig\cdot} \;:\; \{I\; O \;:\; \mathsf{Set}\} \\
&\quad \rightarrow \mathsf{Code}_i\; I\; O \rightarrow (I \rightarrow \mathsf{Code}_{ig}) \rightarrow (O \rightarrow \mathsf{Code}_{ig}) \\
&\text{¡}\Uparrow^{ig\cdot}\; U_i \qquad\quad r\; o \;=\; U_{ig} \\
&\text{¡}\Uparrow^{ig\cdot}\; (I_i\; i) \qquad r\; o \;=\; r\; i \\
&\text{¡}\Uparrow^{ig\cdot}\; (!_i\; i) \qquad r\; o \;=\; K_{ig}\; (o \equiv i) \\
&\text{¡}\Uparrow^{ig\cdot}\; (F \oplus_i G)\; r\; o \;=\; (\text{¡}\Uparrow^{ig\cdot}\; F\; r\; o) \oplus_{ig} (\text{¡}\Uparrow^{ig\cdot}\; G\; r\; o) \\
&\text{¡}\Uparrow^{ig\cdot}\; (F \otimes_i G)\; r\; o \;=\; (\text{¡}\Uparrow^{ig\cdot}\; F\; r\; o) \otimes_{ig} (\text{¡}\Uparrow^{ig\cdot}\; G\; r\; o) \\
&\text{¡}\Uparrow^{ig\cdot}\; (F \circledcirc_i G)\; r\; o \;=\; R_{ig}\; (\sharp\; \text{¡}\Uparrow^{ig\cdot}\; F\; (\text{¡}\Uparrow^{ig\cdot}\; G\; r)\; o) \\
&\text{¡}\Uparrow^{ig\cdot}\; (\mathsf{Fix}_i\; F)\; r\; o \;=\; R_{ig}\; (\sharp\; \text{¡}\Uparrow^{ig\cdot}\; F\; [r\, ,\, \text{¡}\Uparrow^{ig\cdot}\; (\mathsf{Fix}_i\; F)\; r]\; o)
\end{aligned}
$$

Outline

Conclusion

- Analyzed five libraries, with an encoding in Agda for each
- Relations between the fixed-point approaches made (formally) clear
- The same generic function can be used in different libraries
- Embedding into InstantGenerics requires expanding fixed points, highlighting e.g. the way composition works in PolyP

Conclusion

- Analyzed five libraries, with an encoding in Agda for each
- Relations between the fixed-point approaches made (formally) clear
- The same generic function can be used in different libraries
- Embedding into InstantGenerics requires expanding fixed points, highlighting e.g. the way composition works in PolyP

Future work:

- Prove termination, remove `--type-in-type`
- Expand to other generic views