Universiteit Utrecht

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

Formally comparing approaches to datatype-generic programming, using Agda

José Pedro Magalhães
joint work with Andres Löh

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

August 27, 2011

Outline

Setting

- There are many libraries for generic programming in Haskell
- Different approaches vary widely 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 widely 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
    K     : (X : Set)      → Code
    _⊕_   : (F G : Code) → Code
    _⊗_   : (F G : Code) → Code
```

Regular—interpretation

$$\llbracket \_ \rrbracket : \text{Code} \to (\text{Set} \to \text{Set})$$

$$\begin{aligned}
\llbracket\, \mathsf{U} \,\rrbracket\, A &= \top \\
\llbracket\, \mathsf{I} \,\rrbracket\, A &= A \\
\llbracket\, \mathsf{K}\ X \,\rrbracket\, A &= X \\
\llbracket\, F \oplus G \,\rrbracket\, A &= \llbracket\, F \,\rrbracket\, A \uplus \llbracket\, G \,\rrbracket\, A \\
\llbracket\, F \otimes G \,\rrbracket\, A &= \llbracket\, F \,\rrbracket\, A \times \llbracket\, G \,\rrbracket\, A
\end{aligned}$$

data $\mu\ (F : \text{Code}) : \text{Set}$ **where**

$\quad \langle \_ \rangle : \llbracket\, F \,\rrbracket\, (\mu\ F) \to \mu\ F$

Regular—map

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

Outline

PolyP—universe

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

PolyP—interpretation

mutual

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

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

PolyP—map

```
mutual
  map : {A B R S : Set} (F : Code)
        → (A → B) → (R → S) → ⟦ F ⟧ A R → ⟦ F ⟧ B S
  map U        f g _         = tt
  map I        f g x         = g x
  map P        f g x         = f x
  map (K X)    f g x         = x
  map (F ⊕ G) f g (inj₁ x)  = inj₁ (map F f g x)
  map (F ⊕ G) f g (inj₂ x)  = inj₂ (map G f g x)
  map (F ⊗ G) f g (x , y)   = map F f g x , map G f g y
  map (F ⊚ G) f g ⟨ x ⟩     = ⟨ map F (map G f g)
                                   (map (F ⊚ G) f g) x ⟩

  pmap : {A B : Set} (F : Code)
         → (A → B) → μ F A → μ F B
  pmap F f ⟨ x ⟩ = ⟨ map F f (pmap F f) x ⟩
```

Outline

Indexed functors—universe

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

Indexed functors—interpretation

$$
\begin{aligned}
&\mathsf{Indexed} : \mathsf{Set} \to \mathsf{Set}\\
&\mathsf{Indexed}\ I = I \to \mathsf{Set}
\end{aligned}
$$

mutual

$$
\begin{aligned}
&[\![\_]\!] : \{I\ O : \mathsf{Set}\} \to \mathsf{Code}\ I\ O \to \mathsf{Indexed}\ I \to \mathsf{Indexed}\ O\\
&[\![\ U\ ]\!]\ r\ i = \top\\
&[\![\ I\ j\ ]\!]\ r\ i = r\ j\\
&[\![\ !\ j\ ]\!]\ r\ i = i \equiv j\\
&[\![\ F \oplus G\ ]\!]\ r\ i = [\![\ F\ ]\!]\ r\ i \uplus [\![\ G\ ]\!]\ r\ i\\
&[\![\ F \otimes G\ ]\!]\ r\ i = [\![\ F\ ]\!]\ r\ i \times [\![\ G\ ]\!]\ r\ i\\
&[\![\ F \circledcirc G\ ]\!]\ r\ i = [\![\ F\ ]\!]\ ([\![\ G\ ]\!]\ r)\ i\\
&[\![\ \mathsf{Fix}\ F\ ]\!]\ r\ i = \mu\ F\ r\ i
\end{aligned}
$$

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

Indexed functors—map

$$
\begin{aligned}
\_ \rightrightarrows \_ &: \{I : \mathsf{Set}\} \to \mathsf{Indexed}\ I \to \mathsf{Indexed}\ I \to \mathsf{Set} \\
R \rightrightarrows S &= (i : \_) \to R\ i \to S\ i
\end{aligned}
$$

$$
\begin{aligned}
\_\|\_ \ &: \{I\ J : \mathsf{Set}\}\ \{A\ C : \mathsf{Indexed}\ I\}\ \{B\ D : \mathsf{Indexed}\ J\} \\
&\quad \to A \rightrightarrows C \to B \rightrightarrows D \to [A , B] \rightrightarrows [C , D]
\end{aligned}
$$

$$
\begin{aligned}
&\mathsf{map} : \{I\ O : \mathsf{Set}\}\ \{R\ S : \mathsf{Indexed}\ I\}\ (F : \mathsf{Code}\ I\ O) \\
&\qquad \to R \rightrightarrows S \to [\![F]\!]\ R \rightrightarrows [\![F]\!]\ S \\
&\mathsf{map}\ \mathsf{U}\ f\ i\ \_ = \mathsf{tt} \\
&\mathsf{map}\ (\mathsf{I}\ j)\ f\ i\ x = f\ j\ x \\
&\mathsf{map}\ (!\ j)\ f\ i\ x = x \\
&\mathsf{map}\ (F \oplus G)\ f\ i\ (\mathsf{inj}_1\ x) = \mathsf{inj}_1\ (\mathsf{map}\ F\ f\ i\ x) \\
&\mathsf{map}\ (F \oplus G)\ f\ i\ (\mathsf{inj}_2\ x) = \mathsf{inj}_2\ (\mathsf{map}\ G\ f\ i\ x) \\
&\mathsf{map}\ (F \otimes G)\ f\ i\ (x , y) = \mathsf{map}\ F\ f\ i\ x , \mathsf{map}\ G\ f\ i\ y \\
&\mathsf{map}\ (F \circledcirc G)\ f\ i\ x = \mathsf{map}\ F\ (\mathsf{map}\ G\ f)\ i\ x \\
&\mathsf{map}\ (\mathsf{Fix}\ F)\ f\ i\ \langle\ x\ \rangle = \langle\ \mathsf{map}\ F\ (f \,\|\, \mathsf{map}\ (\mathsf{Fix}\ F)\ f)\ i\ x\ \rangle
\end{aligned}
$$

Outline

Instant generics—universe

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


Universiteit Utrecht

[Faculty of Science
Information and Computing Sciences]

Instant generics—interpretation

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

Instant generics—sample generic function

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

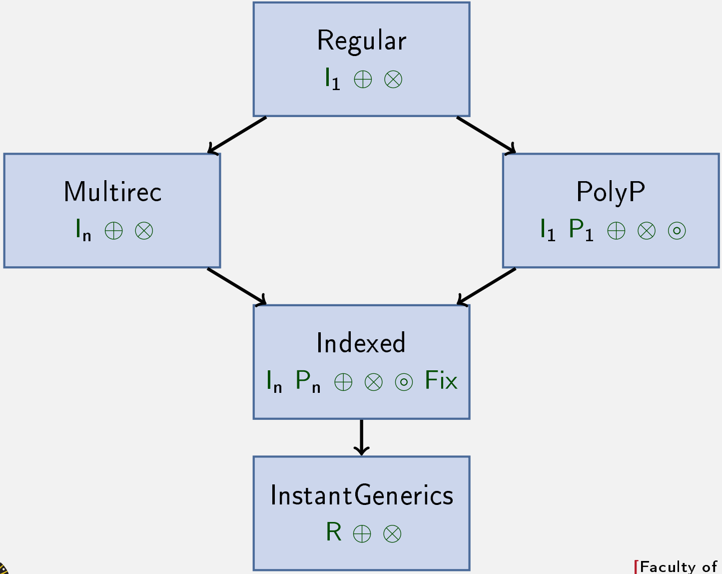
$$
\begin{array}{llll}
\mathsf{size} : (A : \mathsf{Code}) \to [\![A]\!] \to \mathbb{N} & & & \\
\mathsf{size}\ \mathsf{Z} & () & & \\
\mathsf{size}\ \mathsf{U} & x & = & 1 \\
\mathsf{size}\ (\mathsf{K}\ A) & (\mathsf{k}\ x) & = & 1 \\
\mathsf{size}\ (\mathsf{R}\ C) & (\mathsf{rec}\ x) & = & 1 + \mathsf{size}\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

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

module Regular↗PolyP **where**

$$
\begin{array}{lcl}
{}_r{\Uparrow}^p : \mathsf{Code}_r \to \mathsf{Code}_p & & \\
{}_r{\Uparrow}^p\ \mathsf{U}_r & = & \mathsf{U}_p \\
{}_r{\Uparrow}^p\ \mathsf{I}_r & = & \mathsf{I}_p \\
{}_r{\Uparrow}^p\ (\mathsf{K}_r\ X) & = & \mathsf{K}_p\ X \\
{}_r{\Uparrow}^p\ (F \oplus_r G) & = & ({}_r{\Uparrow}^p\ F) \oplus_p ({}_r{\Uparrow}^p\ G) \\
{}_r{\Uparrow}^p\ (F \otimes_r G) & = & ({}_r{\Uparrow}^p\ F) \otimes_p ({}_r{\Uparrow}^p\ G)
\end{array}
$$

$$
\begin{array}{lcl}
{}_r{\uparrow}^p\ :\ \{A\ R\ :\ \mathsf{Set}\} & & \\
\quad \to (C\ :\ \mathsf{Code}_r) \to [\![C]\!]_r\ R \equiv [\![{}_r{\Uparrow}^p\ C]\!]_p\ A\ R & & \\
{}_r{\uparrow}^p\ (\mathsf{U}_r) & = & \mathsf{refl} \\
{}_r{\uparrow}^p\ (\mathsf{I}_r) & = & \mathsf{refl} \\
{}_r{\uparrow}^p\ (\mathsf{K}_r\ X) & = & \mathsf{refl} \\
{}_r{\uparrow}^p\ (F \oplus_r G) & = & \mathsf{cong}_2\ \_\uplus\_\ ({}_r{\uparrow}^p\ F)\ ({}_r{\uparrow}^p\ G) \\
{}_r{\uparrow}^p\ (F \otimes_r G) & = & \mathsf{cong}_2\ \_\times\_\ ({}_r{\uparrow}^p\ F)\ ({}_r{\uparrow}^p\ G)
\end{array}
$$

Regular to PolyP II

$$\text{from}_r : \{A\ R : \text{Set}\}\ (C : \text{Code}_r) \rightarrow [\![C]\!]_r\ R \rightarrow [\![{}_r{\Uparrow}^p\ C]\!]_p\ A\ R$$

$$\begin{aligned}
\text{from}_r\ U_r &= \text{id} \\
\text{from}_r\ I_r &= \text{id} \\
\text{from}_r\ (K_r\ X) &= \text{id} \\
\text{from}_r\ (F \oplus_r G) &= [\text{inj}_1 \circ \text{from}_r\ F, \text{inj}_2 \circ \text{from}_r\ G] \\
\text{from}_r\ (F \otimes_r G) &= < \text{from}_r\ F \circ \text{proj}_1, \text{from}_r\ G \circ \text{proj}_2 >
\end{aligned}$$

$$\text{from}\mu_r : \{A : \text{Set}\}\ (C : \text{Code}_r) \rightarrow \mu_r\ C \rightarrow \mu_p\ ({}_r{\Uparrow}^p\ C)\ A$$

$$\text{from}\mu_r\ C\ \langle x \rangle_r = \langle \text{from}_r\ C\ (\text{map}_r\ C\ (\text{from}\mu_r\ C)\ x) \rangle_p$$

Regular to PolyP III

$$
\begin{array}{lcl}
\mathsf{iso}_1 & : & \{A\ R : \mathsf{Set}\}\ (C : \mathsf{Code}_r)\ \{x : [\![\_]\!]_r\ C\ R\} \\
 & & \rightarrow \mathsf{to}_r\ \{A\}\ C\ (\mathsf{from}_r\ C\ x) \equiv x \\
\mathsf{iso}_1\ \mathsf{U}_r & = & \mathsf{refl} \\
\mathsf{iso}_1\ \mathsf{I}_r & = & \mathsf{refl} \\
\mathsf{iso}_1\ (\mathsf{K}_r\ X) & = & \mathsf{refl} \\
\mathsf{iso}_1\ (F \oplus_r G)\ \{\mathsf{inj}_1\ \_\} & = & \mathsf{cong}\ \mathsf{inj}_1\ (\mathsf{iso}_1\ F) \\
\mathsf{iso}_1\ (F \oplus_r G)\ \{\mathsf{inj}_2\ \_\} & = & \mathsf{cong}\ \mathsf{inj}_2\ (\mathsf{iso}_1\ G) \\
\mathsf{iso}_1\ (F \otimes_r G)\ \{\_\,,\_\} & = & \mathsf{cong}_2\ \_\,,\_\ (\mathsf{iso}_1\ F)\ (\mathsf{iso}_1\ G)
\end{array}
$$

Regular to PolyP IV

$$\mathsf{iso}\mu_1 \;:\; \{\mathsf{A} : \mathsf{Set}\}\;(\mathsf{C} : \mathsf{Code_r})\;(\mathsf{x} : \mu_\mathsf{r}\;\mathsf{C})$$
$$\rightarrow \mathsf{to}\mu_\mathsf{r}\;\{\mathsf{A}\}\;\mathsf{C}\;(\mathsf{from}\mu_\mathsf{r}\;\mathsf{C}\;\mathsf{x}) \equiv \mathsf{x}$$

$$\mathsf{iso}\mu_1\;\{\mathsf{A}\}\;\mathsf{C}\;\langle \mathsf{x} \rangle_\mathsf{r} \;=\; \mathsf{cong}\;\langle\_\rangle_\mathsf{r}\;\$\;\mathsf{begin}$$
$$\mathsf{to_r}\;\{\mathsf{A}\}\;\mathsf{C}\;(\mathsf{map_p}\;({}_\mathsf{r}\!\Uparrow^\mathsf{p}\;\mathsf{C})\;\mathsf{id}\;(\mathsf{to}\mu_\mathsf{r}\;\mathsf{C})$$
$$(\mathsf{from_r}\;\mathsf{C}\;(\mathsf{map_r}\;\mathsf{C}\;(\mathsf{from}\mu_\mathsf{r}\;\mathsf{C})\;\mathsf{x})))$$
$$\equiv\langle\;\mathsf{mc}\;\{\mathsf{A}\}\;\mathsf{C}\;\rangle$$
$$\mathsf{map_r}\;\mathsf{C}\;(\mathsf{to}\mu_\mathsf{r}\;\{\mathsf{A}\}\;\mathsf{C})\;(\mathsf{to_r}\;\{\mathsf{A}\}\;\mathsf{C}$$
$$(\mathsf{from_r}\;\mathsf{C}\;(\mathsf{map_r}\;\mathsf{C}\;(\mathsf{from}\mu_\mathsf{r}\;\mathsf{C})\;\mathsf{x})))$$
$$\equiv\langle\;\mathsf{cong}\;(\mathsf{map_r}\;\mathsf{C}\;(\mathsf{to}\mu_\mathsf{r}\;\{\mathsf{A}\}\;\mathsf{C}))\;(\mathsf{iso}_1\;\mathsf{C})\;\rangle$$
$$\mathsf{map_r}\;\mathsf{C}\;(\mathsf{to}\mu_\mathsf{r}\;\{\mathsf{A}\}\;\mathsf{C})\;(\mathsf{map_r}\;\mathsf{C}\;(\mathsf{from}\mu_\mathsf{r}\;\{\mathsf{A}\}\;\mathsf{C})\;\mathsf{x})$$

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 (↑^p_r 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)))
  ≡⟨ cong (map_r C (toμ_r {A} C)) (iso1 C) ⟩
    map_r C (toμ_r {A} C) (map_r C (fromμ_r {A} C) x)

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

Regular to PolyP V

$$\text{map}_r\ C\ (\text{to}\mu_r\ \{A\}\ C)\ (\text{map}_r\ C\ (\text{from}\mu_r\ \{A\}\ C)\ x)$$

$$\equiv\langle\ \text{map}_r^{\circ}\ C\ \rangle$$

$$\text{map}_r\ C\ (\text{to}\mu_r\ C \circ \text{from}\mu_r\ C)\ x$$

$$\equiv\langle\ \text{map}_r^{\forall}\ C\ (\text{to}\mu_r\ C \circ \text{from}\mu_r\ C)\ \text{id}\ (\text{iso}\mu_1\ C)\ x\ \rangle$$

$$\text{map}_r\ C\ \text{id}\ x$$

$$\equiv\langle\ \text{map}_r^{\text{id}}\ C\ \rangle$$

$$x \quad \square$$

Regular to PolyP V

$$\begin{aligned}
&\quad \mathsf{map_r}\ C\ (\mathsf{to}\mu_r\ \{A\}\ C)\ (\mathsf{map_r}\ C\ (\mathsf{from}\mu_r\ \{A\}\ C)\ x) \\
&\equiv\langle\ \mathsf{map_r^\circ}\ C\ \rangle \\
&\quad \mathsf{map_r}\ C\ (\mathsf{to}\mu_r\ C \circ \mathsf{from}\mu_r\ C)\ x \\
&\equiv\langle\ \mathsf{map_r^\forall}\ C\ (\mathsf{to}\mu_r\ C \circ \mathsf{from}\mu_r\ C)\ \mathsf{id}\ (\mathsf{iso}\mu_1\ C)\ x\ \rangle \\
&\quad \mathsf{map_r}\ C\ \mathsf{id}\ x \\
&\equiv\langle\ \mathsf{map_r^{id}}\ C\ \rangle \\
&\quad x \quad \square
\end{aligned}$$

$$\begin{aligned}
\mathsf{map_r^\forall}\ :\ & \{A\ B\ :\ \mathsf{Set}\}\ (C\ :\ \mathsf{Code_r}) \\
& \to (f\ g\ :\ A \to B) \to (\forall\ x \to f\ x \equiv g\ x) \\
& \to (\forall\ x \to \mathsf{map_r}\ C\ f\ x \equiv \mathsf{map_r}\ C\ g\ x)
\end{aligned}$$

PolyP to InstantGenerics

$$
\begin{array}{l}
\textbf{module}\ \mathsf{PolyP}{\nearrow}\mathsf{InstantGenerics}\ \textbf{where} \\
\quad {}_{p}{\Uparrow}^{ig} : \mathsf{Code}_p \rightarrow \mathsf{Set} \rightarrow \mathsf{Code}_{ig} \\
\quad {}_{p}{\Uparrow}^{ig}\ C\ A\ =\ {}_{p}{\Uparrow}^{ig\cdot}\ C\ C\ A\ \textbf{where} \\
\qquad {}_{p}{\Uparrow}^{ig\cdot} : \mathsf{Code}_p \rightarrow \mathsf{Code}_p \rightarrow \mathsf{Set} \rightarrow \mathsf{Code}_{ig} \\
\qquad {}_{p}{\Uparrow}^{ig\cdot}\ C\ U_p\ A = U_{ig} \\
\qquad {}_{p}{\Uparrow}^{ig\cdot}\ C\ I_p\ A = R_{ig}\ (\sharp\ {}_{p}{\Uparrow}^{ig\cdot}\ C\ C\ A) \\
\qquad {}_{p}{\Uparrow}^{ig\cdot}\ C\ P_p\ A = K_{ig}\ A \\
\qquad {}_{p}{\Uparrow}^{ig\cdot}\ C\ (K_p\ X)\ A = K_{ig}\ X \\
\qquad {}_{p}{\Uparrow}^{ig\cdot}\ C\ (F \oplus_p G)\ A = (\sharp\ {}_{p}{\Uparrow}^{ig\cdot}\ C\ F\ A) \oplus_{ig} (\sharp\ {}_{p}{\Uparrow}^{ig\cdot}\ C\ G\ A) \\
\qquad {}_{p}{\Uparrow}^{ig\cdot}\ C\ (F \otimes_p G)\ A = (\sharp\ {}_{p}{\Uparrow}^{ig\cdot}\ C\ F\ A) \otimes_{ig} (\sharp\ {}_{p}{\Uparrow}^{ig\cdot}\ C\ G\ A) \\
\qquad {}_{p}{\Uparrow}^{ig\cdot}\ C\ (F \circledcirc_p G)\ A = R_{ig}\ (\sharp\ {}_{p}{\Uparrow}^{ig\cdot}\ F\ F\ [\![({}_{p}{\Uparrow}^{ig\cdot}\ C\ G\ A)]\!]_{ig})
\end{array}
$$

Indexed to InstantGenerics

module Indexed↗InstantGenerics **where**

$$
\begin{aligned}
&\mathsf{i2c}'_c \;:\; \{I\;O \;:\; \mathsf{Set}\} \\
&\quad \to \mathsf{Code}_i\;I\;O \to (I \to \mathsf{Code}_{ig}) \to (O \to \mathsf{Code}_{ig}) \\
&\mathsf{i2c}'_c\;U_i\;r\;o = U_{ig} \\
&\mathsf{i2c}'_c\;(I_i\;i)\;r\;o = r\;i \\
&\mathsf{i2c}'_c\;(!_i\;i)\;r\;o = U_{ig} \\
&\mathsf{i2c}'_c\;(F \oplus_i G)\;r\;o = (\sharp\;\mathsf{i2c}'_c\;F\;r\;o) \oplus_{ig} (\sharp\;\mathsf{i2c}'_c\;G\;r\;o) \\
&\mathsf{i2c}'_c\;(F \otimes_i G)\;r\;o = (\sharp\;\mathsf{i2c}'_c\;F\;r\;o) \otimes_{ig} (\sharp\;\mathsf{i2c}'_c\;G\;r\;o) \\
&\mathsf{i2c}'_c\;(F \circledcirc_i G)\;r\;o = R_{ig}\;(\sharp\;\mathsf{i2c}'_c\;F\;(\mathsf{i2c}'_c\;G\;r)\;o) \\
&\mathsf{i2c}'_c\;(\mathsf{Fix}_i\;F)\;r\;o = R_{ig}\;(\sharp\;\mathsf{i2c}'_c\;F\;[r\,,\mathsf{i2c}'_c\;(\mathsf{Fix}_i\;F)\;r]\;o)
\end{aligned}
$$

$$
\begin{aligned}
&\mathsf{i2c}_c \;:\; \{I\;O \;:\; \mathsf{Set}\} \\
&\quad \to \mathsf{Code}_i\;I\;O \to (I \to \mathsf{Set}) \to (O \to \mathsf{Code}_{ig}) \\
&\mathsf{i2c}_c\;C\;r\;o = \mathsf{i2c}'_c\;C\;(K_{ig} \circ r)\;o
\end{aligned}
$$

Universiteit Utrecht

[Faculty of Science
Information and Computing Sciences]

Outline

Introduction

Generic programming libraries, in Agda

Regular

PolyP

Indexed functors

Instant generics

Conversions

Conclusion

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
- InstantGenerics has a very flexible universe
- Embedding into InstantGenerics requires expanding fixed points, highlighting e.g. the way composition works in PolyP