What You Needa Know About Yoneda

Jeremy Gibbons

S-REPLS #12, July 2019

What You Needa Know About Yoneda

Jeremy Gibbons

1. Nobuo Yoneda

- 1930–1996
- Professor of Theoretical Foundations of Information Science, University of Tokyo
- member of IFIP WG2.1, contributor to discussions about Algol 68, major role in Algol N
- but primarily an algebraist

2. The Yoneda Lemma

"Arguably the most important result in category theory" (Emily Riehl).

The *Yoneda Lemma*, formally:

> For $\mathbb{C}$ a locally small category, $[\mathbb{C}, \mathbb{S}et](\mathbb{C}(A, -), F) \simeq F(A)$,
> naturally in $A \in \mathbb{C}$ and $F \in [\mathbb{C}, \mathbb{S}et]$.

From left to right, take nt $\phi : \mathbb{C}(A, -) \stackrel{.}{\to} F$ to $\phi_A(id_A) \in F(A)$.
From right to left, take element $x \in F(A)$ to nt ϕ such that $\phi_B(f) = F(f)(x)$.

As a special case, the *Yoneda Embedding*:

> The functor $Y : \mathbb{C}^{\mathrm{op}} \to [\mathbb{C}, \mathbb{S}et]$ is full and faithful, and injective on objects.

$$[\mathbb{C}, \mathbb{S}et](\mathbb{C}(A, -), \mathbb{C}(B, -)) \simeq \mathbb{C}(B, A)$$

Quite fearsomely terse!

3. The Yoneda Lemma, philosophically

- roughly, “a thing is determined by its relationships with other things”
- in English, *you can tell a lot about a person by the company they keep*
- in Japanese, 人間 (‘ningen’, *human being*) constructed from 人 (‘nin’, *person*) and 間 (‘gen’, *between*),
- in Euclidean geometry, a point ‘is’ just the lines that meet there
- in poetry, *c’est l’exécution du poème qui est le poème* (Valéry)
- in music, *die Idee der Interpretation gehört zur Musik selber und ist ihr nicht akzidentiell* (Adorno)
- in sculpture, need to see a work from all angles in order to understand it (Mazzola)

4. The Yoneda Lemma, computationally

Approximate category $\mathbb{S}et$ by Haskell:

- objects A are types, arrows f are functions
- functors are represented by the type class $Functor$
- natural transformations are polymorphic functions.

Then Yoneda representation $Yo\ f\ a \simeq f\ a$ of functorial datatypes:

$$\mathbf{data}\ Yo\ f\ a = Yo\ \{unYo :: \forall x\,.\,(a \to x) \to f\ x\}$$

$$\begin{array}{ll} fromYo :: Yo\ f\ a \to f\ a & \\ fromYo\ y = unYo\ y\ id & \text{-- apply to identity} \end{array}$$

$$\begin{array}{ll} toYo :: Functor\ f \Rightarrow f\ a \to Yo\ f\ a & \\ toYo\ x = Yo\ (\lambda h \to fmap\ h\ x) & \text{-- use functorial action} \end{array}$$

—“an F of As is a method for yielding an F of Xs, given an $A \to X$”.

5. The Yoneda Lemma, dually

Now consider one half of this bijection:

$$\begin{aligned} & \forall A\,.\,F(A) \to (\forall X\,.\,(A \to X) \to F(X)) \\ \simeq\ & \forall A\,.\,\forall X\,.\,F(A) \to (A \to X) \to F(X) \\ \simeq\ & \forall A\,.\,\forall X\,.\,(F(A) \times (A \to X)) \to F(X) \\ \simeq\ & \forall X\,.\,\forall A\,.\,(F(A) \times (A \to X)) \to F(X) \\ \simeq\ & \forall X\,.\,(\exists A\,.\,F(A) \times (A \to X)) \to F(X) \end{aligned}$$

Hence a kind of dual, ‘co-Yoneda’: $(\exists x\,.\,(x \to a, f\ x)) \simeq f\ a$ for functor f —“an F of Xs can be represented by an F of As and an abstraction $A \to X$”.

Indeed, for functor F and object X:

$$(\exists A\,.\,F(A) \times (A \to X)) \simeq F(X)$$

But even for non-functor F, lhs is functorial. . .

IoT toaster

Signature of commands for remote interaction:

$$
\begin{aligned}
&\mathbf{data}\ \mathit{Command} :: * \to * \quad \mathbf{where} \\
&\quad \mathit{Say} \;\; :: \mathit{String} \to \mathit{Command}\ () \\
&\quad \mathit{Toast} :: \mathit{Int} \quad\;\; \to \mathit{Command}\ () \\
&\quad \mathit{Sense} :: () \quad\;\;\; \to \mathit{Command}\ \mathit{Int}
\end{aligned}
$$

GADT, but not a functor; type *index* rather than type *parameter*.
So no free monad, for representing terms.

Co-Yoneda trick to the rescue:

$$\mathbf{data}\ \mathit{Action}\ a = \exists r\ .\ A\ (\mathit{Command}\ r, r \to a)$$

Action *is* a functor, even though *Command* is not.

6. The Yoneda Lemma, familiarly (1)

Any preorder $(A, \leqslant)$ forms a degenerate category: arrow $a \to b$ iff $a \leqslant b$.

Proof by *indirect inequality* or *indirect order*

$$(b \leqslant a) \Leftrightarrow (\forall c\,.\,(a \leqslant c) \Rightarrow (b \leqslant c))$$

is the Yoneda Embedding in category $\mathbb{P}re(\leqslant)$:

$$[\mathbb{P}re(\leqslant), \mathbb{S}et](\mathbb{P}re(\leqslant)(a, -), \mathbb{P}re(\leqslant)(b, -)) \simeq \mathbb{P}re(\leqslant)(b, -)(a) = \mathbb{P}re(\leqslant)(b, a)$$

- homset $\mathbb{P}re(A, \leqslant)(b, a)$ is a 'thin set': singleton or empty
- homfunctor $\mathbb{P}re(A, \leqslant)(a, -)$ takes $c \in A$ to singleton set when $a \leqslant c$, else empty
- natural transformation $\phi : \mathbb{P}re(A, \leqslant)(a, -) \mathrel{\dot{\to}} \mathbb{P}re(A, \leqslant)(b, -)$ is function family $\phi_c : \mathbb{P}re(A, \leqslant)(a, c) \to \mathbb{P}re(A, \leqslant)(b, c)$
- so ϕ_c is a witness that if $\mathbb{P}re(A, \leqslant)(a, c) \not\equiv \varnothing$ then $\mathbb{P}re(A, \leqslant)(b, c) \not\equiv \varnothing$

The Yoneda Lemma, familiarly (2)

Any functor F *preserves* isos:

$$F(f) \circ F(g) = id \;\Leftrightarrow\; F(f \circ g) = F(id) \;\Leftarrow\; f \circ g = id$$

Full and faithful functor (ie bijection on arrows) also *reflects* isos.

Hom functor $\mathbb{C}(-, =)$ is full and faithful; so

$$(\mathbb{C}(B, -) \simeq \mathbb{C}(A, -)) \;\Leftrightarrow\; (A \simeq B) \;\Leftrightarrow\; (\mathbb{C}(-, A) \simeq \mathbb{C}(-, B))$$

in particular for $\mathbb{C} = \mathbb{P}re(\leqslant)$, the rule of *indirect equality*:

$$(b = a) \;\Leftrightarrow\; (\forall c \,.\, (a \leqslant c) \Leftrightarrow (b \leqslant c))$$

The Yoneda Lemma, familiarly (3)

Cayley's Theorem

every group is isomorphic to a group of bijections

and similarly

every monoid is isomorphic to a monoid of endomorphisms

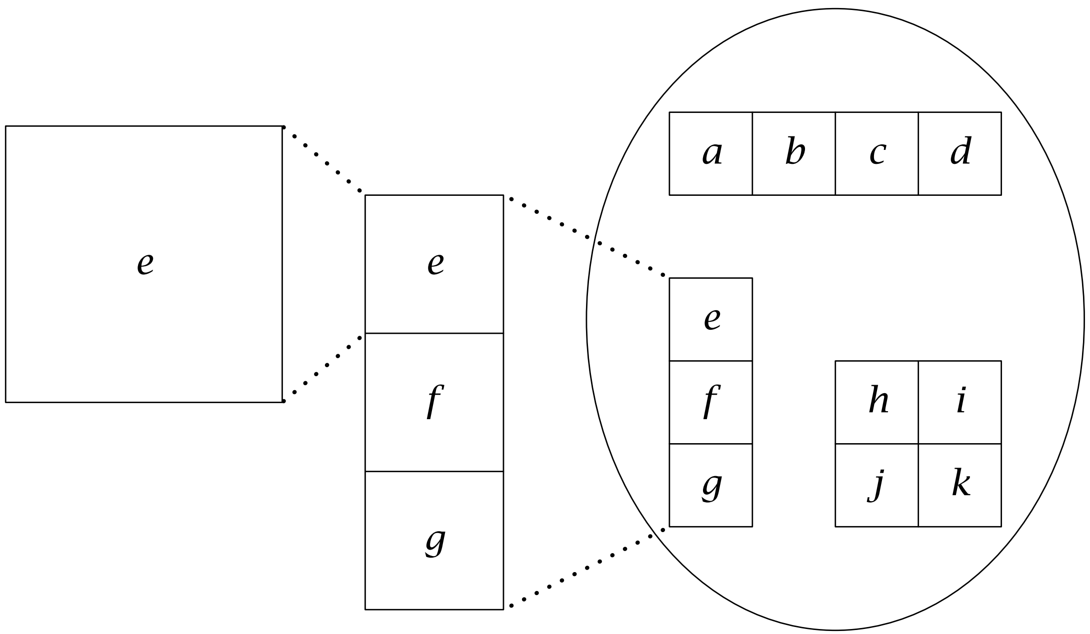
$$(M, \oplus, e) \simeq (\{(m\oplus) \mid m \in M\}, (\circ), id)$$

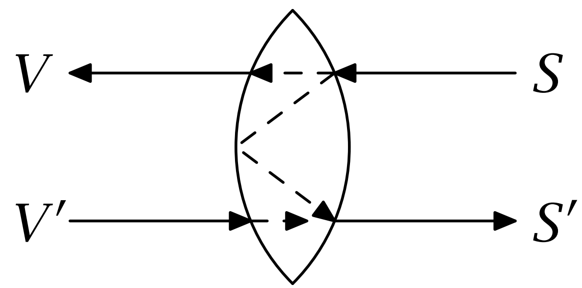
A standard trick, in particular for the monoid $([A], +\!\!+, [\,])$:

represent list xs by function $(xs +\!\!+)$, linear-time $+\!\!+$ by constant-time $\circ$

7. Optics

- *compositional references* (Kagawa, Oles)
- *lens* combinators (Foster, Pierce)





Concrete optics

A view onto a product type:

$$\mathbf{data}\ \mathit{Lens}\ a\ b\ s\ t = \mathit{Lens}\ \{ \begin{aligned}[t] &\mathit{view} &&:: s \to a, \\ &\mathit{update} &&:: s \times b \to t \} \end{aligned}$$

A view onto a sum type:

$$\mathbf{data}\ \mathit{Prism}\ a\ b\ s\ t = \mathit{Prism}\ \{ \begin{aligned}[t] &\mathit{match} &&:: s \to t + a, \\ &\mathit{build} &&:: b \to t \} \end{aligned}$$

Common specialization, for adapting interfaces:

$$\mathbf{data}\ \mathit{Adapter}\ a\ b\ s\ t = \mathit{Adapter}\ \{ \begin{aligned}[t] &\mathit{from} &&:: s \to a, \\ &\mathit{to} &&:: b \to t \} \end{aligned}$$

But they don't compose well—what is a *Lens* composed with a *Prism*?

Profunctor optics

Formally, functors $\mathbb{C}^{\mathrm{op}} \times \mathbb{C} \to \mathbb{S}et$.
Informally, *transformers*, which *consume and produce*:

$$\begin{array}{l} \textbf{class}\ \mathit{Profunctor}\ p\ \textbf{where} \\ \quad \mathit{dimap} :: (c \to a) \to (b \to d) \to p\ a\ b \to p\ c\ d \end{array}$$

Plain functions $\to$ are the canonical instance:

$$\begin{array}{l} \textbf{instance}\ \mathit{Profunctor}\ (\to)\ \textbf{where} \\ \quad \mathit{dimap}\ f\ g\ h = g \circ h \circ f \end{array}$$

Profunctor adapter adapts transformer $p\ a\ b$ to $p\ s\ t$, *uniformly* in p:

$$\textbf{type}\ \mathit{AdapterP}\ a\ b\ s\ t = \forall p\ .\ \mathit{Profunctor}\ p \Rightarrow p\ a\ b \to p\ s\ t$$

Now ordinary functions, so compose straightforwardly.

Similar constructions for lenses and prisms.

Double Yoneda Embedding

Key insight, by Bartosz Milewski, from two applications of Yoneda:

$$[[\mathbb{C}, \mathbb{S}et], \mathbb{S}et]((-)A, (-)B) \simeq \mathbb{C}(A, B)$$

In Haskell, this says:

$$(\forall f\ .\ \mathit{Functor}\ f \Rightarrow f\ a \to f\ b) \simeq (a \to b)$$

Indeed, these are inverses:

$$\begin{array}{ll} \mathit{fromFun} :: (\forall f\ .\ \mathit{Functor}\ f \Rightarrow f\ a \to f\ b) \to (a \to b) & \\ \mathit{fromFun}\ \mathit{phi} = \mathit{unId} \circ \mathit{phi} \circ \mathit{Id} & \text{-- apply to identity} \\ \mathit{toFun} :: (a \to b) \to (\forall f\ .\ \mathit{Functor}\ f \Rightarrow f\ a \to f\ b) & \\ \mathit{toFun} = \mathit{fmap} & \text{-- use functorial action} \end{array}$$

But this works in any category, including $\mathbb{C}^{\mathrm{op}} \times \mathbb{C}$.
Hence $\mathit{AdapterP}\ a\ b\ s\ t \simeq \mathit{Adapter}\ a\ b\ s\ t$. Similarly for lenses and prisms.

8. Conclusions

- Yoneda Lemma is essentially simple, but surprisingly deep
- profunctor optics are practical and powerful, but mysterious
- Yoneda simplifies previously convoluted proofs of equivalence

- some of these ideas due to Guillaume Boisseau, Ed Kmett, Russell O'Connor, Matthew Pickering, Bartosz Milewski

- paper with GB at ICFP 2018