Profunctor optics: a categorical update

Mario Román, Bryce Clarke, Fosco Loregian, Emily Pillmore,
Derek Elkins, Bartosz Milewski and Jeremy Gibbons

September 5, 2019

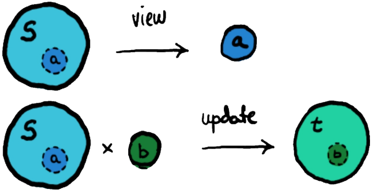
SYCO 5, University of Birmingham

Part 1: Motivation

Lenses

Definition (Oles, 1982)

$$\mathbf{Lens}\left(\begin{pmatrix} a \\ b \end{pmatrix}, \begin{pmatrix} s \\ t \end{pmatrix}\right) = \mathbf{Sets}(s, a) \times \mathbf{Sets}(s \times b, t).$$

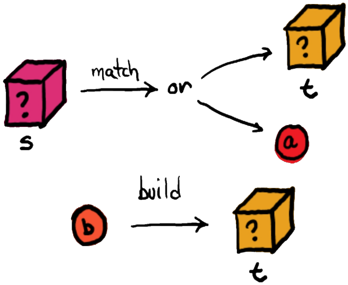


$$(example) \qquad \text{view}\colon \mathbf{Postal} \to \mathbf{Street}$$
$$\text{update}\colon \mathbf{Postal} \times \mathbf{Street} \to \mathbf{Postal}$$

Prisms (alternatives)

Definition

$$\mathbf{Prism}\left(\begin{pmatrix} a \\ b \end{pmatrix}, \begin{pmatrix} s \\ t \end{pmatrix}\right) = \mathbf{Sets}(s, t + a) \times \mathbf{Sets}(b, t).$$

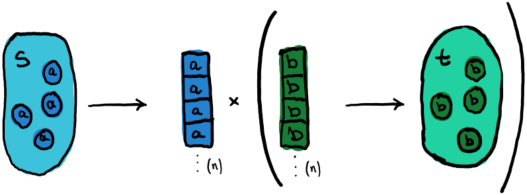


$$(\textit{example}) \qquad \text{match} \colon \mathbf{String} \to \mathbf{String} + \mathbf{Postal}$$
$$\text{build} \colon \mathbf{Postal} \to \mathbf{String}$$

Traversals (and multiple foci)

Definition

$$\mathbf{Traversal}\left(\begin{pmatrix} s \\ t \end{pmatrix}, \begin{pmatrix} a \\ b \end{pmatrix}\right) = \mathbf{Sets}\left(s, \sum_n a^n \times (b^n \to t)\right).$$



$(\textit{example})$ $\qquad \text{extract} \colon \mathbf{MailList} \to \sum_{n \in \mathbb{N}} \mathbf{Email}^n \times (\mathbf{Email}^n \to \mathbf{MailList})$

This is not modular

How to compose two lenses? How to compose a Prism with a Lens?

$$\begin{pmatrix} s \\ t \end{pmatrix} \xrightarrow[\text{prism}]{m,b} \begin{pmatrix} a \\ b \end{pmatrix} \xrightarrow[\text{lens}]{v,u} \begin{pmatrix} x \\ y \end{pmatrix}$$

This is not modular

How to compose two lenses? How to compose a Prism with a Lens?

$$\begin{pmatrix} s \\ t \end{pmatrix} \xrightarrow[\text{prism}]{m,b} \begin{pmatrix} a \\ b \end{pmatrix} \xrightarrow[\text{lens}]{v,u} \begin{pmatrix} x \\ y \end{pmatrix}$$

Every case (Prism+Lens, Lens+Prism, Traversal+Prism+Other...) needs special attention. For instance, a lens and a prism

$$\begin{aligned} \mathbf{v} \in \mathbf{Sets}(s, t + a) \qquad & \qquad \mathbf{m} \in \mathbf{Sets}(a, x) \\ \mathbf{u} \in \mathbf{Sets}(b, t) \qquad & \qquad \mathbf{b} \in \mathbf{Sets}(a \times y, b) \end{aligned}$$

can be composed into the following morphism, which is neither a lens nor a prism.

$$\mathbf{m} \circ [\mathrm{id}_t, \mathbf{v} \times \Lambda(\mathbf{b} \circ \mathbf{u})] \in \mathbf{Sets}(s, t + x \times (y \to t)).$$

This is not modular (in code)

```haskell
-- Given a lens and a prism,
viewStreet   :: Postal -> Street
updateStreet :: Postal -> Street -> Postal
matchAddress :: String -> Either String Postal
buildAddress :: Postal -> String

-- the composition is neither a lens nor a prism.
parseStreet :: String -> Either String (Street , Street -> Postal)
parseStreet s = case matchAddress s of
  Left  addr -> Left  addr
  Right post -> Right (viewStreet post, updateStreet post)
```

Profunctor optics

Perhaps surprisingly, some optics are equivalent to parametric functions over profunctors.

- Lenses are parametric functions.

$$\mathbf{Sets}(s, a) \times \mathbf{Sets}(s \times b, t) \quad \cong \quad \forall p \in \mathrm{Tmb}(\times).p(a, b) \to p(s, t)$$

- Prisms are parametric functions.

$$\mathbf{Sets}(a, a + x) \times \mathbf{Sets}(y, b) \quad \cong \quad \forall p \in \mathrm{Tmb}(+).p(x, y) \to p(a, b)$$

Where $p \in \mathrm{Tmb}(\otimes)$ is called a *Tambara module*; this means we have a natural transformation $p(a, b) \to p(c \otimes a, c \otimes b)$ subject to some conditions

Profunctor optics

Perhaps surprisingly, some optics are equivalent to parametric functions over profunctors.

- Lenses are parametric functions.

$$\mathbf{Sets}(s,a) \times \mathbf{Sets}(s \times b, t) \quad \cong \quad \forall p \in \mathrm{Tmb}(\times).p(a,b) \to p(s,t)$$

- Prisms are parametric functions.

$$\mathbf{Sets}(a, a+x) \times \mathbf{Sets}(y,b) \quad \cong \quad \forall p \in \mathrm{Tmb}(+).p(x,y) \to p(a,b)$$

Where $p \in \mathrm{Tmb}(\otimes)$ is called a *Tambara module*; this means we have a natural transformation $p(a,b) \to p(c \otimes a, c \otimes b)$ subject to some conditions

This solves composition

Now composition of optics is just *function composition*. From $p(a,b) \to p(s,t)$ and $p(x,y) \to p(a,b)$ we can get $p(x,y) \to p(s,t)$.

An example in Haskell

```
-- Haskell code  --



let address = "15 Parks Rd, OX1 3QD, Oxford"

address^.postal
-- Street:  15 Parks Rd
-- Code:    OX1 3QD
-- City:    Oxford

address^.postal.street
-- "15 Parks Rd"

address^.postal.street <~ "7 Banbury Rd"
-- "7 Banbury Rd, OX1 3QD, Oxford"



-------------------
```

Outline

- Existential optics: a definition of optic.
- Profunctor optics: on optics as parametric functions.
- Composing optics: on how composition works.
- Case study: on how to invent an optic.
- Further work: and implementations.

Part 2: Existential optics

(co)Ends

Ends and *Coends* over a profunctor $p\colon \mathbf{C}^{op} \times \mathbf{C} \to \mathbf{Sets}$ are special kinds of (co)limits , (co)equalizing its right and left mapping.

$$\int_{x \in \mathbf{C}} p(x,x) \longrightarrow \prod_{x \in \mathbf{C}} p(x,x) \underset{p(f,\mathrm{id})}{\overset{p(\mathrm{id},f)}{\rightrightarrows}} \prod_{f\colon a \to b} p(a,b)$$

$$\coprod_{f\colon b \to a} p(a,b) \underset{p(f,\mathrm{id})}{\overset{p(\mathrm{id},f)}{\rightrightarrows}} \coprod_{x \in \mathbf{C}} p(x,x) \longrightarrow \int^{x \in \mathbf{C}} p(x,x)$$

Intuitively, a *natural* universal quantifier (ends) and existential quantifier (coends).

Fosco Loregian. "This is the (co)end, my only (co)friend". In: *arXiv preprint arXiv:1501.02503* (2015).

(Co)end calculus

Natural transformations can be rewritten in terms of ends. For any $F, G \colon \mathbf{C} \to \mathbf{D}$,

$$\mathrm{Nat}(F, G) = \int_{x \in \mathbf{C}} \mathbf{D}(Fx, Gx).$$

We can compute (co)ends using the Yoneda lemma.

$$\int_{x \in \mathbf{C}} \mathbf{Sets}(\mathbf{C}(x, a), Gx) \cong Ga,$$

$$\int^{x \in \mathbf{C}} Fx \times \mathbf{C}(a, x) \cong Fa.$$

Continuity of the hom functor takes the following form.

$$\mathbf{D}\left(\int^{c \in \mathbf{C}} p(c, c), d\right) \cong \int_{c \in \mathbf{C}} \mathbf{D}(p(c, c), d),$$

$$\mathbf{D}\left(d, \int_{c \in \mathbf{C}} p(c, c)\right) \cong \int_{c \in \mathbf{C}} \mathbf{D}(d, p(c, c)).$$

(Co)end calculus

Natural transformations can be rewritten in terms of ends. For any $F, G \colon \mathbf{C} \to \mathbf{D}$,

$$\mathrm{Nat}(F, G) = \int_{x \in \mathbf{C}} \mathbf{D}(Fx, Gx).$$

We can compute (co)ends using the Yoneda lemma.

$$\int_{x \in \mathbf{C}} \mathbf{Sets}(\cancelto{x\,=\,a}{\mathbf{C}(x, a)}, Gx) \cong Ga,$$

$$\int^{x \in \mathbf{C}} Fx \times \cancelto{x\,=\,a}{\mathbf{C}(a, x)} \cong Fa.$$

Continuity of the hom functor takes the following form.

$$\mathbf{D}\left(\int^{c \in \mathbf{C}} p(c, c), d\right) \cong \int_{c \in \mathbf{C}} \mathbf{D}(p(c, c), d),$$

$$\mathbf{D}\left(d, \int_{c \in \mathbf{C}} p(c, c)\right) \cong \int_{c \in \mathbf{C}} \mathbf{D}(d, p(c, c)).$$

A definition of "optic"

Definition (Milewski, Boisseau/Gibbons, Riley, generalized)

Fix a monoidal category $\mathbf{M}$ with a strong monoidal functor $\underline{(\_)} \colon \mathbf{M} \to [\mathbf{C}, \mathbf{C}]$. Let $s, t, a, b \in \mathbf{C}$; an optic from (s, t) with *focus* on (a, b) is an element of the following set.

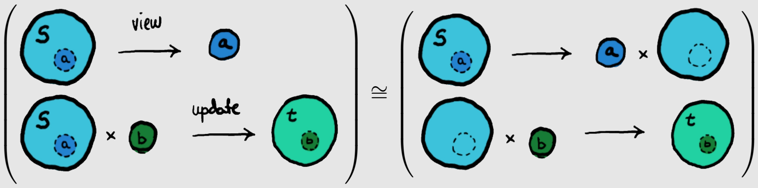
$$\mathbf{Optic}\left(\begin{pmatrix} a \\ b \end{pmatrix}, \begin{pmatrix} s \\ t \end{pmatrix}\right) = \int^{m \in \mathbf{M}} \mathbf{C}(s, \underline{m}a) \times \mathbf{C}(\underline{m}b, t).$$

Intuition: The optic splits into some focus a and some *context* m. We cannot access that context, but we can use it to update.

Lenses are optics

Proposition (from Milewski, 2017)

Lenses are optics for the product.



Proof.

$$\int^{c \in \mathbf{Sets}} \mathbf{Sets}(s, c \times a) \times \mathbf{Sets}(c \times b, t) \cong \qquad \text{(Product)}$$

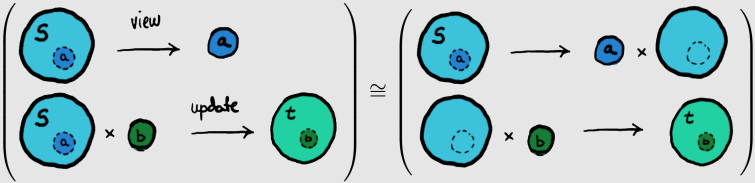
$$\int^{c \in \mathbf{Sets}} \mathbf{Sets}(s, c) \times \mathbf{Sets}(s, a) \times \mathbf{Sets}(c \times b, t) \cong \qquad \text{(Yoneda)}$$

$$\mathbf{Sets}(s, a) \times \mathbf{Sets}(s \times b, t)$$

Lenses are optics

Proposition (from Milewski, 2017)

Lenses are optics for the product.



Proof.

$$\int^{c \in \mathbf{Sets}} \mathbf{Sets}(s, c \times a) \times \mathbf{Sets}(c \times b, t) \cong \qquad (\textit{ Product })$$

$$\int^{c \in \mathbf{Sets}} \mathbf{Sets}(s, c) \times \mathbf{Sets}(s, a) \times \mathbf{Sets}(c \times b, t) \cong \qquad (\textit{ Yoneda })$$

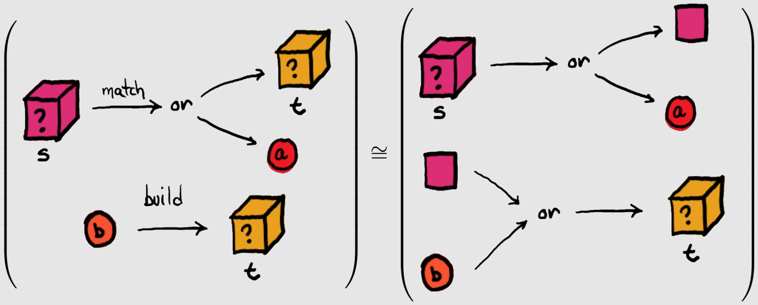
$c = s$

$$\mathbf{Sets}(s, a) \times \mathbf{Sets}(s \times b, t)$$

Prisms are optics

Proposition (Milewski, 2017)

Dually, prisms are optics for the coproduct.



Proof.

$$\int^{m \in \mathbf{Sets}} \mathbf{Sets}(s, m+a) \times \mathbf{Sets}(m+b, t) \cong \quad \text{(Coproduct)}$$

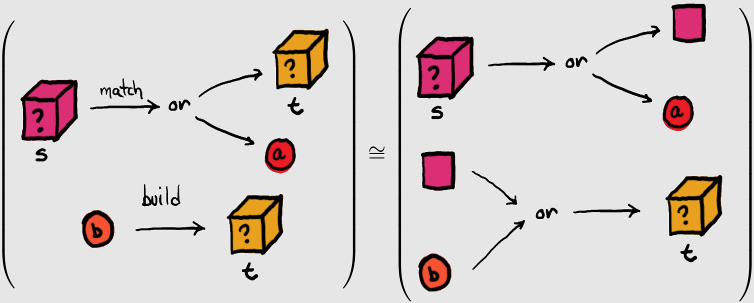
$$\int^{m \in \mathbf{Sets}} \mathbf{Sets}(s, m+a) \times \mathbf{Sets}(m, t) \times \mathbf{Sets}(b, t) \cong \quad \text{(Yoneda)}$$

$$\mathbf{Sets}(s, t+a) \times \mathbf{Sets}(b, t)$$

Prisms are optics

Proposition (Milewski, 2017)

Dually, prisms are optics for the coproduct.



Proof.

$$\int^{m \in \mathbf{Sets}} \mathbf{Sets}(s, m+a) \times \mathbf{Sets}(m+b, t) \cong \quad \text{(Coproduct)}$$

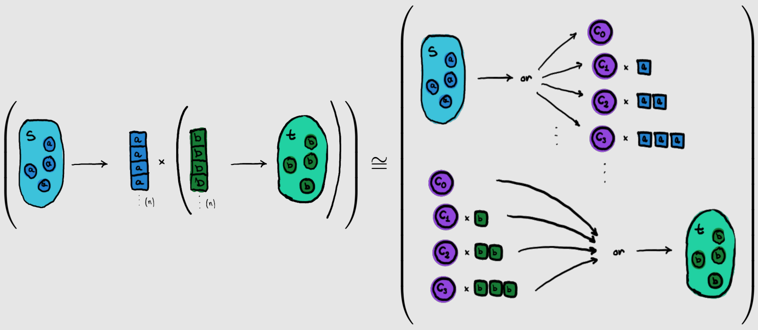
$$\int^{m \in \mathbf{Sets}} \mathbf{Sets}(s, m+a) \times \mathbf{Sets}(m, t) \times \mathbf{Sets}(b, t) \cong \quad \text{(Yoneda)}$$

$m = t$

$$\mathbf{Sets}(s, t+a) \times \mathbf{Sets}(b, t)$$

Theorem

Traversals are optics for the action of polynomial functors $\sum_n c_n \times \square^n$.



That is,

$$\int^{c} \mathbf{Sets}\left(s, \Sigma_n(c_n \times a^n)\right) \times \mathbf{Sets}\left(\Sigma_n(c_n \times b^n), t\right) \cong \mathbf{Sets}(s, \Sigma_n a^n \times (b^n \to t)).$$

Traversals are optics: proof

Again by the Yoneda lemma, this time for functors $c\colon \mathbb{N} \to \mathbf{Sets}$.

$$\int^{c} \mathbf{Sets}\left(s, \sum\nolimits_n c_n \times a^n\right) \times \mathbf{Sets}\left(\sum\nolimits_n c_n \times b^n, t\right) \cong \quad \textit{(cocontinuity)}$$

$$\int^{c} \mathbf{Sets}\left(s, \sum\nolimits_n c_n \times a^n\right) \times \prod_n \mathbf{Sets}\left(c_n \times b^n, t\right) \cong \quad \textit{(prod/exp adjunction)}$$

$$\int^{c} \mathbf{Sets}\left(s, \sum\nolimits_n c_n \times a^n\right) \times \prod_n \mathbf{Sets}\left(c_n, b^n \to t\right) \cong \quad \textit{(natural transf. as an end)}$$

$$\int^{c} \mathbf{Sets}(s, \sum\nolimits_n c_n \times a^n) \times [\mathbb{N}, \mathbf{Sets}]\left(c_{\square}, b^{\square} \to t\right) \cong \quad \textit{(Yoneda lemma)}$$

$$\mathbf{Sets}\left(s, \sum\nolimits_n a^n \times (b^n \to t)\right)$$

Programming libraries use traversable functors to describe traversals. Polynomials are related to these *traversable* functors by a result of Jaskelioff/O'Connor.

Traversals are optics: proof

Again by the Yoneda lemma, this time for functors $c \colon \mathbb{N} \to \mathbf{Sets}$.

$$\int^{c} \mathbf{Sets}\left(s, \sum_n c_n \times a^n\right) \times \mathbf{Sets}\left(\sum_n c_n \times b^n, t\right) \cong \quad \textit{(cocontinuity)}$$

$$\int^{c} \mathbf{Sets}\left(s, \sum_n c_n \times a^n\right) \times \prod_n \mathbf{Sets}\left(c_n \times b^n, t\right) \cong \quad \textit{(prod/exp adjunction)}$$

$$\int^{c} \mathbf{Sets}\left(s, \sum_n c_n \times a^n\right) \times \prod_n \mathbf{Sets}\left(c_n, b^n \to t\right) \cong \quad \textit{(natural transf. as an end)}$$

$$c = b^{\square} \to t$$

$$\int^{c} \mathbf{Sets}(s, \sum_n c_n \times a^n) \times [\mathbb{N}, \mathbf{Sets}]\left(c_{\square}, b^{\square} \to t\right) \cong \quad \textit{(Yoneda lemma)}$$

$$\mathbf{Sets}\left(s, \sum_n a^n \times (b^n \to t)\right)$$

Programming libraries use traversable functors to describe traversals. Polynomials are related to these *traversable* functors by a result of Jaskelioff/O'Connor.

Unification of optics

All the usual optics are of this form. Some new ones arise naturally.

Name	Concrete	Action
Adapter	$(s \to a) \times (b \to t)$	Identity
Lens	$(s \to a) \times (b \times s \to t)$	Product
Prism	$(s \to t + a) \times (b \to t)$	Coproduct
Grate	$((s \to a) \to b) \to t$	Exponential
Affine Traversal	$s \to t + a \times (b \to t)$	Product and coproduct
Glass	$((s \to a) \to b) \to s \to t$	Product and exponential
Traversal	$s \to \Sigma n.a^n \times (b^n \to t)$	Polynomials
Setter	$(a \to b) \to (s \to t)$	Any functor

Part 3: the Profunctor representation theorem

For an action $(\_) \colon \mathbf{M} \to [\mathbf{C}, \mathbf{C}]$.

Tambara modules

Definition (from Pastro/Street)

A Tambara module is a profunctor p together with a family of morphisms satisfying some coherence conditions.

$$p(a,b) \to p(\underline{m}a, \underline{m}b), \qquad m \in \mathbf{M}.$$

Pastro and Street showed they are coalgebras for a comonad.

$$\Theta(p)(a,b) = \int_{m \in \mathbf{M}} p(\underline{m}a, \underline{m}b).$$

Or equivalently, algebras for its left adjoint monad $\Psi \dashv \Theta$.

$$\Psi q(x,y) = \int^{m \in \mathbf{M}} \int^{a,b \in \mathbf{C}} q(a,b) \times \mathbf{C}(\underline{m}a, x) \times \mathbf{C}(y, \underline{m}b)$$

We call Tmb to the Eilenberg-Moore category for the monad, or equivalently, for the adjoint comonad.

Profunctor representation

Theorem (Boisseau/Gibbons)

Optics are functions parametric over Tambara modules.

$$\mathbf{Optic}((a,b),(s,t)) \cong \int_{p \in \mathrm{Tmb}} \mathbf{Sets}(Up(a,b), Up(s,t))$$

In fact, $\mathbf{Optic}$ *is the full subcategory on representable functors of the Kleisli category for* Ψ*.*

Profunctor representation: proof

$$\int_{p \in \mathrm{Tmb}} \mathbf{Sets}(p(a,b), p(s,t)) \cong \quad \text{(Yoneda lemma)}$$

$$\int_{p \in \mathrm{Tmb}} \mathbf{Sets}\left(\mathrm{Nat}(\text{よ}(a,b), Up), Up(s,t)\right) \cong \quad \text{(Free Tambara)}$$

$$\int_{p \in \mathrm{Tmb}} \mathbf{Sets}\left(\mathrm{Tmb}(\Psi\text{よ}(a,b), p), Up(s,t)\right) \cong \quad \text{(Yoneda lemma)}$$

$$\Psi\text{よ}(a,b)(s,t) \cong \quad \text{(Definition of } \Psi\text{)}$$

$$\int^{m \in \mathbf{M}} \int^{x,y \in \mathbf{C}} \mathbf{C}(s, \underline{m}x) \times \mathbf{C}(\underline{m}y, t) \times \text{よ}(a,b)(x,y) \cong \quad \text{(Yoneda lemma)}$$

$$\int^{m \in \mathbf{M}} \mathbf{C}(s, \underline{m}a) \times \mathbf{C}(\underline{m}b, t)$$

Because $\Psi\text{よ}(a,b)(s,t) \cong \mathrm{Nat}(\text{よ}(s,t), \Psi\text{よ}(a,b))$, the category of optics is the full subcategory on representable functors of the Kleisli category for Ψ.

Part 3: the Profunctor representation theorem

For an action $(\_) \colon \mathbf{M} \to [\mathbf{C}, \mathbf{C}]$.

(This time in Prof!)

The bicategory Prof

The bicategory $\mathbf{Prof}$ has

- 0-cells are (small) categories $\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D}, \ldots$, as in $\mathbf{Cat}$;
- 1-cells $\mathbf{C} \nrightarrow \mathbf{D}$ are profunctors $p \colon \mathbf{C}^{op} \times \mathbf{D} \to \mathbf{Sets}$,
- 2-cells $p \Rightarrow q$ are natural transformations.

Two profunctors $p \colon \mathbf{C} \nrightarrow \mathbf{D}$ and $q \colon \mathbf{D} \nrightarrow \mathbf{E}$ are composed into $(q \diamond p) \colon \mathbf{C} \nrightarrow \mathbf{E}$ with the following (co)end.

$$(q \diamond p)(c, e) \quad = \quad \int^{d \in \mathbf{D}} \quad p(c, d) \quad \times \quad q(d, e).$$

Yoneda lemma makes the hom profunctor よ$\colon \mathbf{C}^{op} \times \mathbf{C} \to \mathbf{Sets}$ the identity.

The bicategory Prof

The bicategory $\mathbf{Prof}$ has

- 0-cells are (small) categories $\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D}, \ldots$, as in $\mathbf{Cat}$;
- 1-cells $\mathbf{C} \nrightarrow \mathbf{D}$ are profunctors $p \colon \mathbf{C}^{op} \times \mathbf{D} \to \mathbf{Sets}$,
- 2-cells $p \Rightarrow q$ are natural transformations.

Two profunctors $p \colon \mathbf{C} \nrightarrow \mathbf{D}$ and $q \colon \mathbf{D} \nrightarrow \mathbf{E}$ are composed into $(q \diamond p) \colon \mathbf{C} \nrightarrow \mathbf{E}$ with the following (co)end.

$$\begin{array}{ccccccc} (q \diamond p)(c,e) & = & \int^{d \in \mathbf{D}} & p(c,d) & \times & q(d,e). \\ (Q \circ P)(c,e) & \iff & \exists d \in \mathbf{D}. & P(c,d) & \wedge & Q(d,e). \end{array}$$

Yoneda lemma makes the hom profunctor よ$\colon \mathbf{C}^{op} \times \mathbf{C} \to \mathbf{Sets}$ the identity.

Promonads and the optics category

A promonad $\psi \in [\mathbf{A}^{op} \times \mathbf{B}, \mathbf{Sets}]$ is a monoid in the bicategory of profunctors.

Lemma (Kleisli construction in Prof, e.g. in Pastro/Street)

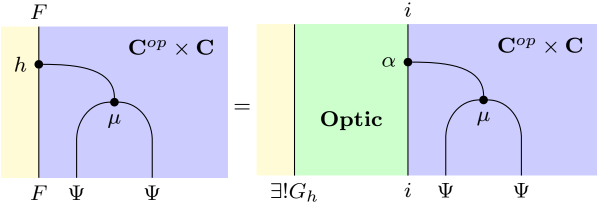
The Kleisli object for the promonad, $\mathrm{Kl}(\psi)$, is a category with the same objects, but hom-sets given by the promonad, $\mathrm{Kl}(\psi)(a,b) = \psi(a,b)$.

For some fixed kind of optic, we can create a category with the same objects as $\mathbf{C}^{op} \times \mathbf{C}$, but where morphisms are optics of that kind.

$$\psi((s,t),(a,b)) = \int^{m \in \mathbf{M}} \mathbf{C}(s, \underline{m}a) \times \mathbf{D}(\underline{m}b, t)$$

That is, $\mathbf{Optic} := \mathrm{Kl}(\psi)$.

Kleisli object



Theorem (Pastro/Street)

Functors $[\mathbf{Optic}, \mathbf{Set}]$ are equivalent to right modules on the terminal object for the promonad $\mathrm{Mod}(\psi)$, which are algebras for an associated monad.

It follows from the universal property of the Kleisli object that

$$\mathbf{Cat}(\mathbf{Optic}, \mathbf{Set}) \cong \mathbf{Prof}(1, \mathbf{Optic}) \cong \mathbf{Mod}(\psi).$$

I am using Dan Marsden's macros for diagrams

Profunctor representation theorem

Theorem (Riley 2018, Boisseau/Gibbons 2018, different proof technique)

Optics given by ψ correspond to parametric functions over profunctors that have (pro)module structure over ψ.

$$\mathbf{Optic}((a,b),(s,t)) \cong \int_{p \in \mathbf{Mod}(\psi)} p(a,b) \to p(s,t)$$

Proof.

$$\begin{aligned}
\int_{p \in \mathbf{Mod}(\psi)} p(a,b) \to p(s,t) &\cong && \textit{(lemma)} \\
\int_{p \in [\mathbf{Optic},\mathbf{Sets}]} p(a,b) \to p(s,t) &\cong && \textit{(by definition)} \\
\mathrm{Nat}(-(a,b), -(s,t)) &\cong && \textit{(Yoneda embedding)} \\
\mathrm{Nat}(\mathrm{Nat}(\mathbf{Optic}((a,b),\Box),-), \mathrm{Nat}(\mathbf{Optic}((s,t),\Box),-)) &\cong && \textit{(Yoneda embedding)} \\
\mathrm{Nat}(\mathbf{Optic}((a,b),\Box), \mathbf{Optic}((s,t),\Box)) &\cong && \textit{(Yoneda embedding)} \\
\mathbf{Optic}((s,t),(a,b))
\end{aligned}$$

Summary

- $\mathbf{Optic}$ is the full subcategory on representable functors of a Kleisli category.
  - In Prof, it is a Kleisli object.
- Tambara modules are algebras for the monad.
  - In Prof, they are (pro)algebras for the promonad. It follows that $[\mathbf{Optic}, \mathbf{Sets}] \cong \mathrm{Tmb}$.

Part 4: Composition of optics

How Haskell composes optics

Given two optics for two actions $\alpha\colon \mathbf{M} \to [\mathbf{C},\mathbf{C}]$ and $\beta\colon \mathbf{N} \to [\mathbf{C},\mathbf{C}]$.

$$\int_{p \in \mathrm{Tmb}(\alpha)} \mathbf{Sets}(p(a,b), p(s,t)), \qquad \int_{q \in \mathrm{Tmb}(\beta)} \mathbf{Sets}(q(x,y), q(a,b)).$$

We can *compose* them into a function polymorphic over profunctors that are algebras for both monads.

$$\int_{(p,p) \in \mathrm{Tmb}(\alpha) \times_{\mathbf{Prof}} \mathrm{Tmb}(\beta)} \mathbf{Sets}(p(a,b), p(s,t))$$

In other words, we consider the following pullback.

$$\begin{array}{ccc} \mathrm{Tmb}(\alpha) \times_{\mathbf{Prof}} \mathrm{Tmb}(\beta) & \xrightarrow{\pi} & \mathrm{Tmb}(\alpha) \\ {\scriptstyle\pi}\downarrow & & \downarrow{\scriptstyle U} \\ \mathrm{Tmb}(\beta) & \xrightarrow{U} & \mathbf{Prof} \end{array}$$

The coproduct comonad

Lemma

A pair of Tambara modules $\mathrm{Tmb}(\alpha)$ *and* $\mathrm{Tmb}(\beta)$ *over the same profunctor* p *is the same as a Tambara modules* $\mathrm{Tmb}(\alpha + \beta)$ *for the coproduct action* $\alpha + \beta \colon \mathbf{M} + \mathbf{N} \to [\mathbf{C}, \mathbf{C}]$.

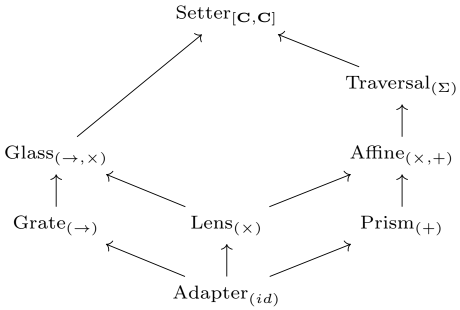
For instance, Haskell would compose lenses and prisms into optics for an action of the following form.

$$a \mapsto c_1 + d_1 \times (c_2 + d_2 \times \dots a)$$

This is usually *projected* into an action of the following form $a \mapsto c + d \times a$ (replete image of the action) that gives an optic called affine traversal.

Lattice of actions

With some notion of subcategory of endofunctors (replete subcategories and pseudomonic functors), we can limit actions to submonoidal categories of $[\mathbf{C}, \mathbf{C}]$.



Part 5: A case study

Applicative functors

Let $(\mathbf{M}, \otimes, i)$ be a monoidal category. $[\mathbf{M}, \mathbf{Sets}]$ is monoidal with Day convolution.

$$(F * G)(m) = \int^{x,y\in\mathbf{M}} \mathbf{M}(x \otimes y, m) \times F(x) \times G(y)$$

Monoids for Day convolution are lax monoidal functors. We can compute free lax monoidal functors as we compute free monoids.

$$F^* = \mathrm{id} + F + F * F + F * F * F + \dots$$

Lax monoidal functors for $\mathbf{Sets}$ are called applicative functors [McBride/Paterson].

The optic for applicative functors

$$\int^{F \in \mathbf{App}} \mathbf{Sets}(s, Fa) \times \mathbf{Sets}(Fb, t)$$

$\cong$ (Yoneda lemma)

$$\int^{F \in \mathbf{App}} \mathrm{Nat}\left(s \times (a \to (-)), F\right) \times \mathbf{Sets}(Fb, t)$$

$\cong$ (Free-forgetful adjunction for applicative functors)

$$\int^{F \in \mathbf{App}} \mathbf{App}\left(\sum_n s^n \times (a^n \to (-))\,, F\right) \times \mathbf{Sets}(Fb, t)$$

$\cong$ (Yoneda lemma)

$$\mathbf{Sets}\left(\sum_n s^n \times (a^n \to b), t\right)$$

$\cong$ (Continuity)

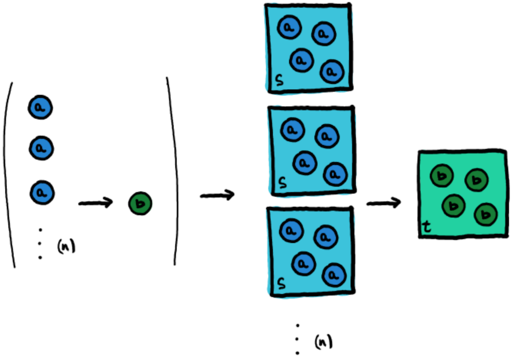
$$\prod_n \mathbf{Sets}\left(s^n \times (a^n \to b), t\right).$$

This can be done in general for any functors that can be generated (co)freely.

Kaleidoscopes

Kaleidoscopes are optics for the evaluation of applicative functors , $\mathbf{App} \to [\mathbf{Sets}, \mathbf{Sets}]$. They have a concrete description

$$\mathbf{Kaleidoscope}\left(\begin{pmatrix} a \\ b \end{pmatrix}, \begin{pmatrix} s \\ t \end{pmatrix}\right) = \prod_n \mathbf{Sets}\left(s^n \times (a^n \to b), t\right).$$



List-lenses

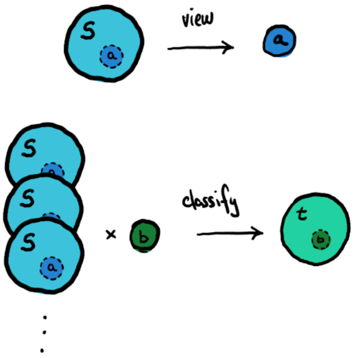
Kaleidoscopes cannot be composed with lenses because $(c \times -)$ is not lax monoidal. It is lax monoidal when c is a monoid. We can ask the residual to be a monoid.

$$
\begin{aligned}
&\int^{c \in \mathsf{Mon}} \mathbf{C}(s, c \times a) \times \mathbf{C}(c \times b, t) && \cong && \text{(Product)} \\
&\int^{c \in \mathsf{Mon}} \mathbf{C}(s, c) \times \mathbf{C}(s, a) \times \mathbf{C}(c \times b, t) && \cong && \text{(Free monoid)} \\
&\int^{c \in \mathsf{Mon}} \mathbf{Mon}(s^*, c) \times \mathbf{C}(s, a) \times \mathbf{C}(c \times b, t) && \cong && \text{(Yoneda lemma)} \\
&\mathbf{C}(s, a) \times \mathbf{C}(s^* \times b, t)
\end{aligned}
$$

List-lenses

List lenses are optics for the product by a monoid , $(\times)\colon \mathbf{Mon} \times \mathbf{Sets} \to \mathbf{Sets}$. They have a concrete description

$$\mathbf{ListLens}\left(\begin{pmatrix} a \\ b \end{pmatrix}, \begin{pmatrix} s \\ t \end{pmatrix}\right) = \mathbf{Sets}(s, a) \times \mathbf{Sets}(s^{*} \times b, t)$$



List-lenses (unlike general lenses) compose with Kaleidoscopes!

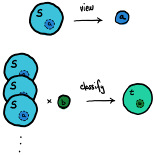
Example

Take the iris dataset. Each entry is a $\mathbf{Flower}$ given by a species and four real numbers $\mathbf{Flower} = \mathbf{Species} \times \mathbb{R}_+^4$.

```
5.1,3.5,1.4,0.2,Iris-setosa
4.9,3.0,1.4,0.2,Iris-setosa
4.7,3.2,1.3,0.2,Iris-setosa
4.6,3.1,1.5,0.2,Iris-setosa
5.0,3.6,1.4,0.2,Iris-setosa
...
```

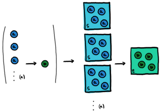
We define a list-lens that implements some learning algorithm.

$$\mathbf{Flower} \to \mathbb{R}_+^4$$
$$\mathbf{Flower}^* \times \mathbb{R}_+^4 \to \mathbf{Flower}$$



We define a Kaleidoscope that takes an aggregating function on $\mathbb{R}_+$ and induces a componentwise aggregating function on the 4-tuples $\mathbb{R}_+^4$.

$$\prod_{n\in\mathbb{N}} \left((\mathbb{R}_+)^n \to \mathbb{R}_+\right) \to \left((\mathbb{R}_+^4)^n \to \mathbb{R}_+^4\right)$$



Example

List-lenses are, in particular, lenses; we can use them to *view* the measurements of the first element of our dataset.

```
(iris !! 1)^.measurements

---- Output ----
   Sepal length: 4.9
   Sepal width:  3.0
   Petal length: 1.4
   Petal width:  0.2
```

Example

They are more abstract than a lens in the sense that they can be used to classify some measurements into a new species taking into account all the examples of the dataset.

```
iris ?. measurements (Measurements 4.8 3.1 1.5 0.1)

---- Output ----
Flower:
    Sepal length: 4.8
    Sepal width:  3.1
    Petal length: 1.5
    Petal width:  0.1
    Species:      Iris setosa  -- <<<< Clasifies the species.
```

Example

List-lenses can be composed with kaleidoscopes. The composition takes an aggregation function and classifies the result

```
iris >- measurements.aggregateWith mean

---- Output ----
Flower:
    Sepal length: 5.843
    Sepal width:  3.054
    Petal length: 3.758
    Petal width:  1.198
    Species:      Iris versicolor
```

Part 4: Summary and further work

Summary

- Optics: a zoo of accessors used by programmers *[Kmett, lens library, 2012]*.
  - Concrete representation: each one is described by some functions.
  - Existential representation: unified definition of optics as a coend.
  - Going from concrete to existential cannot be done in general, we look for some way of eliminating the coend.
- Profunctor optics: for monoidal actions *[Pastro/Street, 2008]*, *[Milewski, 2017]* and general actions *[Boisseau/Gibbons, 2018]*.
  - Profunctor representation: can be composed easily.
  - Going from existential to profunctor and back is done in general.
- Composition of optics: what do we get when composing two optics.
  - Distributive laws is the obvious choice.
  - In Haskell, we consider coproducts of monads.
  - We get a lattice of optics.

Related and further work

- Lawful optics. Studied by *[Riley, 2018]*.
  - Programmers use lawful optics, optics with certain properties.
- Generalizations: in which other settings do we get useful results?
  - Enrichments over a cartesian Benabou cosmos $\mathcal{V}$.
  - We have extended the theorems for *mixed optics*.
- Implementation: developing libraries of optics.
  - A concise library in Haskell. `https://github.com/mroman42/vitrea/`
  - Derivations in Agda / Idris allow us to extract translation algorithms for optics. Everything we have been doing is constructive.

```
lensDerivation {s} {t} {a} {b} =
  begin
   ((∫exists c ∈ Set , ((s -> c × a) × (c × b -> t))))         ≅⟨ ≅-coend (λ c -> trivial) ⟩
   ((∫exists c ∈ Set , (((s -> c) × (s -> a)) × (c × b -> t)))) ≅⟨ ≅-coend (λ c -> trivial) ⟩
   ((∫exists c ∈ Set , ((s -> c) × (s -> a) × (c × b -> t))))   ≅⟨ yoneda ⟩
   ((s -> a) × (s × b -> t))
  qed
```

Some literature

Oles, 1982. *A category theoretic approach to the semantics of programming languages (PhD thesis).* Defines lenses for the first time.

Kmett, 2012. *Lens library.* Implements optics in Haskell.

Pickerings/Gibbons/Wu, 2016. *Profunctor optics: modular data accessors.* Derives lenses, prisms, adapters and traversals in Haskell.

Milewski, 2017. *Profunctor optics, the categorical view.* Tambara modules for lenses and prisms.

Boisseau/Gibbons, 2018. *What you needa know about Yoneda.* General definition of optics and a general profunctor representation theorem. Traversal as the optic for traversables.

Riley, 2018. *Categories of optics.* General framework for obtaining laws for the optics.