Dependent Optics

Pietro Vertechi - Applied Category Theory 2022

Structure

- Brief introduction to lenses.

Structure

- Brief introduction to lenses.
- Functor lenses (in particular, dependent lenses).

Structure

- Brief introduction to lenses.
- Functor lenses (in particular, dependent lenses).
- Lenses for reverse-mode automatic differentiation.

Structure

- Brief introduction to lenses.
- Functor lenses (in particular, dependent lenses).
- Lenses for reverse-mode automatic differentiation.
- Definition and basic properties of (dependent) optics.

Structure

- Brief introduction to lenses.
- Functor lenses (in particular, dependent lenses).
- Lenses for reverse-mode automatic differentiation.
- Definition and basic properties of (dependent) optics.
- Dependent optics and functor lenses comparison.

Structure

- Brief introduction to lenses.
- Functor lenses (in particular, dependent lenses).
- Lenses for reverse-mode automatic differentiation.
- Definition and basic properties of (dependent) optics.
- Dependent optics and functor lenses comparison.
- Representations of dependent optics.

Lenses

Lenses are a useful abstraction to access and update data structures.

```
jane = (name="Jane", age=32)

# get value for a given field
get_age(person) = person.age

# build new structure from old structure and new field value
put_age(person, age) = (name=person.name, age=age)
```

Lenses

Lenses are a useful abstraction to access and update data structures.

```
jane = (name="Jane", age=32)

# get value for a given field
get_age(person) = person.age

# build new structure from old structure and new field value
put_age(person, age) = (name=person.name, age=age)
```

Bidirectionality. We have two methods in *different directions:*

$$\text{get}\colon X \to Y \qquad\qquad \text{put}\colon X \times Y' \to X'.$$

Lenses

Lenses are a useful abstraction to access and update data structures.

```
jane = (name="Jane", age=32)

# get value for a given field
get_age(person) = person.age

# build new structure from old structure and new field value
put_age(person, age) = (name=person.name, age=age)
```

Bidirectionality. We have two methods in *different directions:*

$$\mathrm{get}\colon X \to Y \qquad \mathrm{put}\colon X \times Y' \to X'.$$

Composability. Lenses can be composed to handle nested data:

$$\begin{aligned} X &\to Z & X \times Z' &\to X' \\ x &\mapsto \mathrm{get}_2(\mathrm{get}_1(x)) & (x, z') &\mapsto \mathrm{put}_1(x, \mathrm{put}_2(\mathrm{get}_1(x), z')) \end{aligned}$$

Functor lenses

Richer framework for lenses recently put forward (Spivak, 2019).

Setting. Let $\mathcal{C}$ be a category and $\mathcal{F} \colon \mathcal{C}^{\mathrm{op}} \to \mathbf{Cat}$ a pseudofunctor.

Functor lenses

Richer framework for lenses recently put forward (Spivak, 2019).

Setting. Let $\mathcal{C}$ be a category and $\mathcal{F}\colon \mathcal{C}^{\mathrm{op}} \to \mathbf{Cat}$ a pseudofunctor.

Definition. The category $\mathbf{Lens}_{\mathcal{F}}$ has

- objects of the form $\binom{P}{X}$, where $X \in \mathrm{Ob}(\mathcal{C})$ and $P \in \mathrm{Ob}(\mathcal{F}^X)$,
- morphisms given by

$$\mathbf{Lens}_{\mathcal{F}}\left(\begin{pmatrix}P\\X\end{pmatrix}, \begin{pmatrix}Q\\Y\end{pmatrix}\right) = \coprod_{f\colon X \to Y} \mathcal{F}^X(f^*(Q), P).$$

Here, the notation f^* is a shorthand for $\mathcal{F}(f)$.

Functor lenses

Richer framework for lenses recently put forward (Spivak, 2019).

Setting. Let $\mathcal{C}$ be a category and $\mathcal{F}\colon \mathcal{C}^{\mathrm{op}} \to \mathbf{Cat}$ a pseudofunctor.

Definition. The category $\mathbf{Lens}_{\mathcal{F}}$ has

- objects of the form $\binom{P}{X}$, where $X \in \mathrm{Ob}(\mathcal{C})$ and $P \in \mathrm{Ob}(\mathcal{F}^X)$,
- morphisms given by

$$\mathbf{Lens}_{\mathcal{F}}\left(\begin{pmatrix}P\\X\end{pmatrix}, \begin{pmatrix}Q\\Y\end{pmatrix}\right) = \coprod_{f\colon X \to Y} \mathcal{F}^X(f^*(Q), P).$$

Here, the notation f^* is a shorthand for $\mathcal{F}(f)$.

Bidirectionality. $f\colon X \to Y$ is *forward* and $f^\#\colon f^*(Q) \to P$ is *backward*.

Composability. By functoriality of f^* (and $\mathcal{F}$), functor lenses compose.

Examples of functor lenses

Plain lenses. Functor lenses for $X \mapsto \mathrm{coKleisli(X \times -)}$.

$$\mathbf{Lens}((X, X'), (Y, Y')) = \coprod_{X \to Y} \mathcal{C}(X \times Y', X')$$
$$\simeq \mathcal{C}(X, Y) \times \mathcal{C}(X \times Y', X').$$

Examples of functor lenses

Plain lenses. Functor lenses for $X \mapsto \mathrm{coKleisli}(\mathrm{X} \times -)$.

$$\begin{aligned}\mathbf{Lens}((X, X'), (Y, Y')) &= \coprod_{X \to Y} \mathcal{C}(X \times Y', X') \\ &\simeq \mathcal{C}(X, Y) \times \mathcal{C}(X \times Y', X').\end{aligned}$$

Dependent lenses. Functor lenses for $\mathcal{C}/-$ (contravariant slice functor).

$$\mathbf{DLens}(U \to X, V \to Y) = \coprod_{X \to Y} \mathcal{C}/X(X \times_Y V, U).$$

Examples of functor lenses

Plain lenses. Functor lenses for $X \mapsto \mathbf{coKleisli}(\mathrm{X} \times -)$.

$$\begin{aligned}\mathbf{Lens}((X, X'), (Y, Y')) &= \coprod_{X \to Y} \mathcal{C}(X \times Y', X') \\ &\simeq \mathcal{C}(X, Y) \times \mathcal{C}(X \times Y', X').\end{aligned}$$

Dependent lenses. Functor lenses for $\mathcal{C}/-$ (contravariant slice functor).

$$\mathbf{DLens}(U \to X, V \to Y) = \coprod_{X \to Y} \mathcal{C}/X(X \times_Y V, U).$$

The functor $(X, X') \mapsto (X \times X' \to X)$ embeds lenses inside dependent lenses.

We are replacing trivial bundles $X \times X' \to X$ with general bundles (aka dependent types) $U \to X$.

Reverse-mode automatic differentiation

Aim. Pull back cotangent vectors along a smooth function (dual of the differential).

Reverse-mode automatic differentiation

Aim. Pull back cotangent vectors along a smooth function (dual of the differential).

Procedure.

- Hard-code the derivative of *primitive functions*.
- Apply the chain rule for composition of primitive functions.

Reverse-mode automatic differentiation

Aim. Pull back cotangent vectors along a smooth function (dual of the differential).

Procedure.

- Hard-code the derivative of *primitive functions*.
- Apply the chain rule for composition of primitive functions.

Applications. Compute gradients in deep learning (backpropagation).

Reverse-mode automatic differentiation

Aim. Pull back cotangent vectors along a smooth function (dual of the differential).

Procedure.

- Hard-code the derivative of *primitive functions*.
- Apply the chain rule for composition of primitive functions.

Applications. Compute gradients in deep learning (backpropagation).

Bidirectionality and composability suggest that lenses might be a good fit.

Reverse-mode automatic differentiation

Let $f\colon X \to Y$ be a smooth function between Euclidean spaces.
Then, we can define a lens as follows ($v \in Y^*$ is a dual vector):

$$\mathrm{forward}(x) = f(x) \quad \text{and} \quad \mathrm{backward}(x, v) = (Df(x)^*)v.$$

In other words,

- the $\mathrm{forward}$ method is given by the function,
- the $\mathrm{backward}$ method is given by the dual of the differential.

Reverse-mode automatic differentiation

Let $f\colon X \to Y$ be a smooth function between Euclidean spaces.
Then, we can define a lens as follows ($v \in Y^*$ is a dual vector):

$$\mathrm{forward}(x) = f(x) \quad \text{and} \quad \mathrm{backward}(x, v) = (Df(x)^*)v.$$

In other words,

- the $\mathrm{forward}$ method is given by the function,
- the $\mathrm{backward}$ method is given by the dual of the differential.

Bidirectionality. Value and derivatives computed in opposite directions.

Composability. Lens composition corresponds to the chain rule.

Reverse-mode automatic differentiation

Let $f: X \to Y$ be a smooth function between Euclidean spaces.
Then, we can define a lens as follows ($v \in Y^*$ is a dual vector):

$$\mathrm{forward}(x) = f(x) \quad \text{and} \quad \mathrm{backward}(x, v) = (Df(x)^*)v.$$

In other words,

- the $\mathrm{forward}$ method is given by the function,
- the $\mathrm{backward}$ method is given by the dual of the differential.

Bidirectionality. Value and derivatives computed in opposite directions.

Composability. Lens composition corresponds to the chain rule.

Inefficiency. The forward and backward passes often share computation.

Shared computation between forward and reverse passes

Closure encoding. Combine the forward and backward methods as

$$\begin{aligned} X &\to Y \times [Y', X'] \\ x &\mapsto (f(x),\ v \mapsto (Df(x)^*)v)\,. \end{aligned}$$

Shared computation between forward and reverse passes

Closure encoding. Combine the forward and backward methods as

$$\begin{aligned} X &\to Y \times [Y', X'] \\ x &\mapsto (f(x),\ v \mapsto (Df(x)^*)v)\,. \end{aligned}$$

Advantages.

- Closure can be optimized based on the forward pass computation.
- Default implementation of most automatic differentiation libraries.

Shared computation between forward and reverse passes

Closure encoding. Combine the forward and backward methods as

$$\begin{aligned} X &\to Y \times [Y', X'] \\ x &\mapsto (f(x),\ v \mapsto (Df(x)^*)v)\,. \end{aligned}$$

Advantages.

- Closure can be optimized based on the forward pass computation.
- Default implementation of most automatic differentiation libraries.

Issues.

- No clear representation of shared computation (hidden in a closure).
- We actually need dependent lenses in general, but the equivalent of $X \to Y \times [Y', X']$ for dependent lenses is complicated.

Optics

Let M be a space of *shared* data (between forward and backward passes).

$$l\colon X \to M \times Y \quad \text{and} \quad r\colon M \times Y' \to X'.$$

Optics

Let M be a space of *shared* data (between forward and backward passes).

$$l\colon X \to M \times Y \quad \text{and} \quad r\colon M \times Y' \to X'.$$

Key intuition. If $\mathcal{C}$ is a category with finite products,

$$\int^{M \in \mathcal{C}} \mathcal{C}(X, M \times Y) \times \mathcal{C}(M \times Y', X') \simeq \mathcal{C}(X, Y) \times \mathcal{C}(X \times Y', X').$$

Optics

Let M be a space of *shared* data (between forward and backward passes).

$$l\colon X \to M \times Y \quad \text{and} \quad r\colon M \times Y' \to X'.$$

Key intuition. If $\mathcal{C}$ is a category with finite products,

$$\int^{M \in \mathcal{C}} \mathcal{C}(X, M \times Y) \times \mathcal{C}(M \times Y', X') \simeq \mathcal{C}(X, Y) \times \mathcal{C}(X \times Y', X').$$

A (Cartesian) optic is an equivalence class of a pair of maps

$$l\colon X \to M \times Y \quad \text{and} \quad r\colon M \times Y' \to X'.$$

Optics

Let M be a space of *shared* data (between forward and backward passes).

$$l\colon X \to M \times Y \quad \text{and} \quad r\colon M \times Y' \to X'.$$

Key intuition. If $\mathcal{C}$ is a category with finite products,

$$\int^{M \in \mathcal{C}} \mathcal{C}(X, M \times Y) \times \mathcal{C}(M \times Y', X') \simeq \mathcal{C}(X, Y) \times \mathcal{C}(X \times Y', X').$$

A (Cartesian) optic is an equivalence class of a pair of maps

$$l\colon X \to M \times Y \quad \text{and} \quad r\colon M \times Y' \to X'.$$

Efficiency. The morphism r can read data from M instead of recomputing it (Diffractor.jl).

Optics

The Cartesian product can be replaced with a general symmetric monoidal structure

$$\int^{M \in \mathcal{C}} \mathcal{C}(X, M \otimes Y) \times \mathcal{C}(M \otimes Y', Y).$$

Riley (2018) proved that such optics compose and form a symmetric monoidal category.

Optics

The Cartesian product can be replaced with a general symmetric monoidal structure

$$\int^{M \in \mathcal{C}} \mathcal{C}(X, M \otimes Y) \times \mathcal{C}(M \otimes Y', Y).$$

Riley (2018) proved that such optics compose and form a symmetric monoidal category.

Clarke et al. (2020) described an even more general version, *mixed optics:*

$$\int^{M \in \mathcal{M}} \mathcal{C}_L(X, M \cdot_{L} Y) \times \mathcal{C}_R(M \cdot_{R} Y', Y),$$

where $\cdot_L, \cdot_R$ are *actegories:* actions of a monoidal category $\mathcal{M}$ on $\mathcal{C}_L, \mathcal{C}_R$ respectively.

Optics

The Cartesian product can be replaced with a general symmetric monoidal structure

$$\int^{M\in\mathcal{C}} \mathcal{C}(X, M\otimes Y)\times\mathcal{C}(M\otimes Y', Y).$$

Riley (2018) proved that such optics compose and form a symmetric monoidal category.

Clarke et al. (2020) described an even more general version, *mixed optics:*

$$\int^{M\in\mathcal{M}} \mathcal{C}_L(X, M\cdot_{L} Y)\times\mathcal{C}_R(M\cdot_{R} Y', Y),$$

where $\cdot_L, \cdot_R$ are *actegories:* actions of a monoidal category $\mathcal{M}$ on $\mathcal{C}_L, \mathcal{C}_R$ respectively.

Issue. Unfortunately, this is not sufficient to generalize functor lenses,
where one has a pseudofunctor to $\mathbf{Cat}$ rather than an actegory.

Optics

The Cartesian product can be replaced with a general symmetric monoidal structure

$$\int^{M \in \mathcal{C}} \mathcal{C}(X, M \otimes Y) \times \mathcal{C}(M \otimes Y', Y).$$

Riley (2018) proved that such optics compose and form a symmetric monoidal category.

Clarke et al. (2020) described an even more general version, *mixed optics:*

$$\int^{M \in \mathcal{M}} \mathcal{C}_L(X, M \cdot_{L} Y) \times \mathcal{C}_R(M \cdot_{R} Y', Y),$$

where $\cdot_L, \cdot_R$ are *actegories:* actions of a monoidal category $\mathcal{M}$ on $\mathcal{C}_L, \mathcal{C}_R$ respectively.

Issue. Unfortunately, this is not sufficient to generalize functor lenses,
where one has a pseudofunctor to $\mathbf{Cat}$ rather than an actegory.

Key intuition. An actegory is a pseudofunctor from a bicategory with one object to $\mathbf{Cat}$.
Milewski (2022), Vertechi (2022), and Capucci (2022) allow an arbitrary source bicategory.

The category of dependent optics

Setting. Let $\mathcal{B}$ be a bicategory. Let $\mathcal{L}, \mathcal{R} \colon \mathcal{B}^{\text{op}} \rightrightarrows \mathbf{Cat}$ be pseudofunctors.

The category of dependent optics

Setting. Let $\mathcal{B}$ be a bicategory. Let $\mathcal{L}, \mathcal{R} \colon \mathcal{B}^{\mathrm{op}} \rightrightarrows \mathbf{Cat}$ be pseudofunctors.

Definition. $\mathbf{Optic}_{\mathcal{L},\mathcal{R}}$ is the category with

- objects of the form $(X, X')^A$, where $A \in \mathrm{Ob}(\mathcal{B})$, $X \in \mathrm{Ob}(\mathcal{L}^A)$, and $X' \in \mathrm{Ob}(\mathcal{R}^A)$,
- morphisms given by

$$\mathbf{Optic}_{\mathcal{L},\mathcal{R}}((X, X')^A, (Y, Y')^B) = \int^{f \in \mathcal{B}(A,B)} \mathcal{L}^A(X, f^*Y) \times \mathcal{R}^A(f^{*'}Y', X').$$

Here, the notations $f^*, f^{*'}$ are shorthands for $\mathcal{L}(f), \mathcal{R}(f)$ respectively.

The category of dependent optics

Setting. Let $\mathcal{B}$ be a bicategory. Let $\mathcal{L}, \mathcal{R} \colon \mathcal{B}^{\mathrm{op}} \rightrightarrows \mathbf{Cat}$ be pseudofunctors.

Definition. $\mathbf{Optic}_{\mathcal{L},\mathcal{R}}$ is the category with

- objects of the form $(X, X')^A$, where $A \in \mathrm{Ob}(\mathcal{B})$, $X \in \mathrm{Ob}(\mathcal{L}^A)$, and $X' \in \mathrm{Ob}(\mathcal{R}^A)$,
- morphisms given by

$$\mathbf{Optic}_{\mathcal{L},\mathcal{R}}((X, X')^A, (Y, Y')^B) = \int^{f \in \mathcal{B}(A,B)} \mathcal{L}^A(X, f^*Y) \times \mathcal{R}^A(f^{*'}Y', X').$$

Here, the notations $f^*, f^{*'}$ are shorthands for $\mathcal{L}(f), \mathcal{R}(f)$ respectively.

Bidirectionality. $\mathcal{L}, \mathcal{R}$ encode forward and backward directions.

Composability. By functoriality of $f^*, f^{*'}$ (and $\mathcal{L}, \mathcal{R}$), we can compose dependent optics.

The category of dependent optics

Setting. Let $\mathcal{B}$ be a bicategory. Let $\mathcal{L}, \mathcal{R} \colon \mathcal{B}^{\mathrm{op}} \rightrightarrows \mathbf{Cat}$ be pseudofunctors.

Definition. $\mathbf{Optic}_{\mathcal{L},\mathcal{R}}$ is the category with

- objects of the form $(X, X')^A$, where $A \in \mathrm{Ob}(\mathcal{B})$, $X \in \mathrm{Ob}(\mathcal{L}^A)$, and $X' \in \mathrm{Ob}(\mathcal{R}^A)$,
- morphisms given by

$$\mathbf{Optic}_{\mathcal{L},\mathcal{R}}((X, X')^A, (Y, Y')^B) = \int^{f \in \mathcal{B}(A,B)} \mathcal{L}^A(X, f^* Y) \times \mathcal{R}^A(f^{*'} Y', X').$$

Here, the notations $f^*, f^{*'}$ are shorthands for $\mathcal{L}(f), \mathcal{R}(f)$ respectively.

Bidirectionality. $\mathcal{L}, \mathcal{R}$ encode forward and backward directions.

Composability. By functoriality of $f^*, f^{*'}$ (and $\mathcal{L}, \mathcal{R}$), we can compose dependent optics.

Theorem. $\mathbf{Optic}_{\mathcal{L},\mathcal{R}}$ is a category.

The category of dependent optics

$$\mathbf{Optic}_{\mathcal{L},\mathcal{R}}((X, X')^A, (Y, Y')^B) = \int^{f \in \mathcal{B}(A,B)} \mathcal{L}^A(X, f^* Y) \times \mathcal{R}^A(f^{*'} Y', X')$$

The category of dependent optics

$$\mathbf{Optic}_{\mathcal{L},\mathcal{R}}((X,X')^A,(Y,Y')^B) = \int^{f\in\mathcal{B}(A,B)} \mathcal{L}^A(X,f^*Y)\times\mathcal{R}^A(f^{*'}Y',X')$$

Specializations.

- $\mathcal{B}$ is the delooping of a monoidal category $\Rightarrow$ mixed optics.
- $\mathcal{B}$ is a 1-category and $\mathcal{L}$ is trivial $\Rightarrow$ functor lenses.

The category of dependent optics

$$\mathbf{Optic}_{\mathcal{L},\mathcal{R}}((X,X')^A,(Y,Y')^B) = \int^{f\in\mathcal{B}(A,B)} \mathcal{L}^A(X, f^*Y) \times \mathcal{R}^A(f^{*'}Y', X')$$

Specializations.

- $\mathcal{B}$ is the delooping of a monoidal category $\Rightarrow$ mixed optics.
- $\mathcal{B}$ is a 1-category and $\mathcal{L}$ is trivial $\Rightarrow$ functor lenses.

Properties.

☑ **Coproducts.** If $\mathcal{B}$ has finite coproducts which are turned into products by $\mathcal{L}, \mathcal{R}$, then the category $\mathbf{Optic}_{\mathcal{L},\mathcal{R}}$ has finite coproducts.

The category of dependent optics

$$\mathbf{Optic}_{\mathcal{L},\mathcal{R}}((X,X')^A,(Y,Y')^B) = \int^{f \in \mathcal{B}(A,B)} \mathcal{L}^A(X, f^*Y) \times \mathcal{R}^A(f^{*'}Y', X')$$

Specializations.

- $\mathcal{B}$ is the delooping of a monoidal category $\Rightarrow$ mixed optics.
- $\mathcal{B}$ is a 1-category and $\mathcal{L}$ is trivial $\Rightarrow$ functor lenses.

Properties.

☑ **Coproducts.** If $\mathcal{B}$ has finite coproducts which are turned into products by $\mathcal{L}, \mathcal{R}$, then the category $\mathbf{Optic}_{\mathcal{L},\mathcal{R}}$ has finite coproducts.

☒ **Fibration.** There is in general no fibration $\mathbf{Optic}_{\mathcal{L},\mathcal{R}} \to \mathcal{B}$, due to the equivalence relation induced by coend. We need the bicategory of optics, as in Braithwaite et. al (2021).

The category of dependent optics

$$\mathbf{Optic}_{\mathcal{L},\mathcal{R}}((X,X')^A,(Y,Y')^B) = \int^{f\in\mathcal{B}(A,B)} \mathcal{L}^A(X,f^*Y)\times\mathcal{R}^A(f^{*'}Y',X')$$

Specializations.

- $\mathcal{B}$ is the delooping of a monoidal category $\Rightarrow$ mixed optics.
- $\mathcal{B}$ is a 1-category and $\mathcal{L}$ is trivial $\Rightarrow$ functor lenses.

Properties.

☑ **Coproducts.** If $\mathcal{B}$ has finite coproducts which are turned into products by $\mathcal{L},\mathcal{R}$, then the category $\mathbf{Optic}_{\mathcal{L},\mathcal{R}}$ has finite coproducts.

☒ **Fibration.** There is in general no fibration $\mathbf{Optic}_{\mathcal{L},\mathcal{R}} \to \mathcal{B}$, due to the equivalence relation induced by coend. We need the bicategory of optics, as in Braithwaite et. al (2021).

☐ **Monoidal structure.** Monoidality result for functor lenses may be valid here too.

Functor lenses as dependent optics, take 2

- With trivial $\mathcal{L}$ and only trivial 2-morphisms in $\mathcal{B}$, we can't use f to share computation.
- f becomes, in a sense, the forward part of the optic.
- We will see how to circumvent this issue for dependent lenses.

Functor lenses as dependent optics, take 2

- With trivial $\mathcal{L}$ and only trivial 2-morphisms in $\mathcal{B}$, we can't use f to share computation.
- f becomes, in a sense, the forward part of the optic.
- We will see how to circumvent this issue for dependent lenses.

Lenses as optics.

$$\int^{M \in \mathcal{C}} \mathcal{C}(X, M \times Y) \times \mathcal{C}(M \times Y', Y) \simeq \mathcal{C}(X, Y) \times \mathcal{C}(X \times Y', Y).$$

Functor lenses as dependent optics, take 2

- With trivial $\mathcal{L}$ and only trivial 2-morphisms in $\mathcal{B}$, we can't use f to share computation.
- f becomes, in a sense, the forward part of the optic.
- We will see how to circumvent this issue for dependent lenses.

Lenses as optics.

$$\int^{M \in \mathcal{C}} \mathcal{C}(X, M \times Y) \times \mathcal{C}(M \times Y', Y) \simeq \mathcal{C}(X, Y) \times \mathcal{C}(X \times Y', Y).$$

Dependent lenses as optics. Replace product with fibered product:

$$\int^{M \in \mathcal{C}/(A \times B)} \mathcal{C}/A(X, M \times_B Y) \times \mathcal{C}/A(M \times_B Y', X') \simeq \coprod_{X \to Y} \mathcal{C}/A(X \times_B Y', X').$$

The RHS is the set of dependent lenses from $(X \times_A X') \to X$ to $(Y \times_B Y') \to Y$.

Philosophical FAQs on functor lenses as dependent optics

Is it problematic that different, natural choices of pseudofunctors give rise to the same category of dependent lenses? Which is the most natural?

There is a unique *1-category* of dependent lenses, but at least two reasonable *bicategories* of dependent lenses.

The two choices of pseudofunctors reflect this.

As another example, $\mathbf{Optic}_{(\mathcal{C},\times)}$ and $\mathbf{Lens}_{(\mathcal{C},\times)}$ are not really the same: they are different as bicategories.
Here, we recover them *separately*.

Philosophical FAQs on functor lenses as dependent optics

Is it problematic that different, natural choices of pseudofunctors give rise to the same category of dependent lenses? Which is the most natural?

There is a unique *1-category* of dependent lenses, but at least two reasonable *bicategories* of dependent lenses.

The two choices of pseudofunctors reflect this.

As another example, $\mathbf{Optic}_{(\mathcal{C},\times)}$ and $\mathbf{Lens}_{(\mathcal{C},\times)}$ are not really the same: they are different as bicategories.
Here, we recover them *separately*.

What is the forward part of the optic if $\mathcal{L}$ is trivial?

In the *bicategory* of optics, 1-morphisms are given by a coproduct rather than a coend.

$$\coprod_{f \in \mathcal{B}(A,B)} \mathcal{L}^A(X, f^*Y) \times \mathcal{R}^A(f^{*'}Y', X')$$

$$\downarrow$$

$$\coprod_{f \in \mathcal{B}(A,B)} \mathcal{L}^A(X, f^*Y).$$

Intuitively, the image of a 1-morphism is its forward part, whereas its value within a fiber is the backward part.

User-facing APIs

How should libraries based on dependent optics (e.g., Diffractor.jl) interface with users?

Dependent optics are encoded as equivalence classes of pairs of maps

$$l\colon X \to f^*Y \quad \text{and} \quad r\colon {f^*}'Y' \to X'.$$

The morphism f is somewhat ill-defined (due to the equivalence class).

How can it be hidden from the user?

User-facing APIs

How should libraries based on dependent optics (e.g., Diffractor.jl) interface with users?

Dependent optics are encoded as equivalence classes of pairs of maps

$$l\colon X \to f^*Y \quad \text{and} \quad r\colon f^{*'}Y' \to X'.$$

The morphism f is somewhat ill-defined (due to the equivalence class).

How can it be hidden from the user?

1. **Direct computation.** Compute coend explicitly (if possible) and use that as interface.

User-facing APIs

How should libraries based on dependent optics (e.g., Diffractor.jl) interface with users?

Dependent optics are encoded as equivalence classes of pairs of maps

$$l\colon X \to f^*Y \quad \text{and} \quad r\colon f^{*'}Y' \to X'.$$

The morphism f is somewhat ill-defined (due to the equivalence class).

How can it be hidden from the user?

1. **Direct computation.** Compute coend explicitly (if possible) and use that as interface.
2. **Optics representation.** Define a functor from the category of optics to a *friendlier* category.

Tambara representations

Let $\mathcal{D}$ be an arbitrary category.

Tambara representation. A $\mathcal{D}$-valued Tambara representation consists of

- a functor $P^A \colon \left(\mathcal{L}^A\right)^{\mathrm{op}} \times \mathcal{R}^A \to \mathcal{D}$, for all object A in $\mathcal{B}$,
- a natural transformation $\zeta_f \colon P^B(-, =) \Rightarrow P^A(f^* -, f^{*'} =)$ for all $f \colon A \to B$,

where ζ_f is extranatural in f and respects some coherence laws.

Tambara representations

Let $\mathcal{D}$ be an arbitrary category.

Tambara representation. A $\mathcal{D}$-valued Tambara representation consists of

- a functor $P^A \colon \left(\mathcal{L}^A\right)^{\mathrm{op}} \times \mathcal{R}^A \to \mathcal{D}$, for all object A in $\mathcal{B}$,
- a natural transformation $\zeta_f \colon P^B(-, =) \Rightarrow P^A(f^* -, f^{*'} =)$ for all $f \colon A \to B$,

where ζ_f is extranatural in f and respects some coherence laws.

Theorem. A functor $\mathbf{Optic}^{\mathrm{op}}_{\mathcal{L},\mathcal{R}} \to \mathcal{D}$ is the same as a $\mathcal{D}$-valued Tambara representation.

Tambara representations

Let $\mathcal{D}$ be an arbitrary category.

Tambara representation. A $\mathcal{D}$-valued Tambara representation consists of

- a functor $P^A\colon \left(\mathcal{L}^A\right)^{\mathrm{op}} \times \mathcal{R}^A \to \mathcal{D}$, for all object A in $\mathcal{B}$,
- a natural transformation $\zeta_f\colon P^B(-, =) \Rightarrow P^A(f^*-, f^{*'} =)$ for all $f\colon A \to B$,

where ζ_f is extranatural in f and respects some coherence laws.

Theorem. A functor $\mathbf{Optic}_{\mathcal{L},\mathcal{R}}^{\mathrm{op}} \to \mathcal{D}$ is the same as a $\mathcal{D}$-valued Tambara representation.

Example. In the case of dependent lenses, we have a $\mathbf{Set}$-valued Tambara representation

$$(X, X')^A \mapsto \mathcal{C}/A(X, X').$$

Geometric intuition. In the reverse-mode automatic differentiation case,
it corresponds to the pullback of differential 1-forms along smooth maps.

Conclusions

☑ Dependent optics simultaneously generalize mixed optics and functor lenses.

Conclusions

☑ Dependent optics simultaneously generalize mixed optics and functor lenses.

☑ Dependent optics can be used to efficiently implement bidirectional transformations.

Conclusions

☑ Dependent optics simultaneously generalize mixed optics and functor lenses.

☑ Dependent optics can be used to efficiently implement bidirectional transformations.

☑ Under some simple conditions, the category of dependent optics has finite coproducts.

Conclusions

☑ Dependent optics simultaneously generalize mixed optics and functor lenses.

☑ Dependent optics can be used to efficiently implement bidirectional transformations.

☑ Under some simple conditions, the category of dependent optics has finite coproducts.

☒ To generalize the functor lenses fibration, we need a *bicategory* of dependent optics.

Conclusions

- ☑ Dependent optics simultaneously generalize mixed optics and functor lenses.
- ☑ Dependent optics can be used to efficiently implement bidirectional transformations.
- ☑ Under some simple conditions, the category of dependent optics has finite coproducts.
- ☒ To generalize the functor lenses fibration, we need a *bicategory* of dependent optics.
- ☐ Work remains to be done to establish a monoidality theorem for dependent optics.

Conclusions

☑ Dependent optics simultaneously generalize mixed optics and functor lenses.

☑ Dependent optics can be used to efficiently implement bidirectional transformations.

☑ Under some simple conditions, the category of dependent optics has finite coproducts.

☒ To generalize the functor lenses fibration, we need a *bicategory* of dependent optics.

☐ Work remains to be done to establish a monoidality theorem for dependent optics.

☑ Dependent optics representations have a simple explicit description (Tambara representations).

Conclusions

☑ Dependent optics simultaneously generalize mixed optics and functor lenses.

☑ Dependent optics can be used to efficiently implement bidirectional transformations.

☑ Under some simple conditions, the category of dependent optics has finite coproducts.

☒ To generalize the functor lenses fibration, we need a *bicategory* of dependent optics.

☐ Work remains to be done to establish a monoidality theorem for dependent optics.

☑ Dependent optics representations have a simple explicit description (Tambara representations).

☐ What can one say about Tambara representations for special cases of dependent optics?

Thank you!

Preprint.

Dependent Optics – Vertechi, 2022.

Available at arXiv:2204.09547.

Acknowledgments.

Mattia G. Bergomi

Keno Fischer

Bartosz Milewski