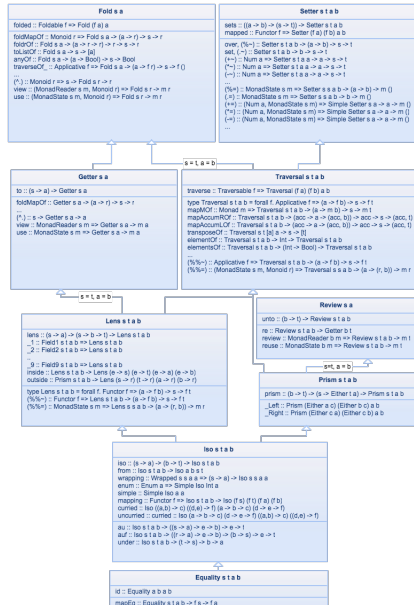


Optics and Type Equivalences

Jacques Carette, Amr Sabry



Based on a 'reversible' core:

Iso s t a b

```
iso :: (s -> a) -> (b -> t) -> Iso s t a b
from :: Iso s t a b -> Iso a b s t
wrapping :: Wrapped s s a a => (s -> a) -> Iso s s a a
enum :: Enum a => Simple Iso Int a
simple :: Simple Iso a a
mapping :: Functor f => Iso s t a b -> Iso (f s) (f t) (f a) (f b)
curried :: Iso ((a,b) -> c) ((d,e) -> f) (a -> b -> c) (d -> e -> f)
uncurried :: curried :: Iso (a -> b -> c) (d -> e -> f) ((a,b) -> c) ((d,e) -> f)

au :: Iso s t a b -> ((s -> a) -> e -> b) -> e -> t
auf :: Iso s t a b -> ((r -> a) -> e -> b) -> (b -> s) -> e -> t
under :: Iso s t a b -> (t -> s) -> b -> a
```

Lens in Haskell

```
data Lens s a = Lens { view  :: s -> a, set   :: s -> a -> s }
```

Lens in Haskell

```
data Lens s a = Lens { view :: s -> a, set :: s -> a -> s }
```

Example:

```
_1 :: Lens (a , b) a  
_1 = Lens { view = fst , set = \s a -> (a, snd b) }
```

Lens in Haskell

```
data Lens s a = Lens { view :: s -> a, set :: s -> a -> s }
```

Example:

```
_1 :: Lens (a , b) a  
_1 = Lens { view = fst , set = \s a -> (a, snd b) }
```

Laws? Optimizations?

```
view (set s a) == a  
set s (view s) == s  
set (set s a) a' == set s a'
```

Lens in Agda

record **GS-Lens** { $ls\ \ell a : \text{Level}$ } ($S : \text{Set } ls$) ($A : \text{Set } \ell a$) : **Set** ($ls \sqcup \ell a$) **where**
 field

get : $S \rightarrow A$

set : $S \rightarrow A \rightarrow S$

getput : { $s : S$ } { $a : A$ } \rightarrow **get** (**set** $s\ a$) $\equiv a$

putget : ($s : S$) \rightarrow **set** s (**get** s) $\equiv s$

putput : ($s : S$) ($a\ a' : A$) \rightarrow **set** (**set** $s\ a$) $a' \equiv$ **set** $s\ a'$

open **GS-Lens**

Works... but the proofs can be tedious in larger examples (later)

fst : { $A\ B : \text{Set}$ } \rightarrow **GS-Lens** ($A \times B$) A

fst = **record** { **get** = **proj**₁
 ; **set** = $\lambda\ s\ a \rightarrow (a , \text{proj}_2\ s)$
 ; **getput** = **refl**
 ; **putget** = $\lambda\ _ \rightarrow$ **refl**
 ; **putput** = $\lambda\ _ _ _ \rightarrow$ **refl** }

Lens in Agda 2

Or, the return of constant-complement lenses:

```
record Lens1 {ℓ : Level} (S : Set ℓ) (A : Set ℓ) : Set (suc ℓ) where
  constructor ∃-lens
  field
    {C} : Set ℓ
    iso : S ≃ (C × A)
```


Lens in Agda 2

Or, the return of constant-complement lenses:

```
record Lens1 {ℓ : Level} (S : Set ℓ) (A : Set ℓ) : Set (suc ℓ) where
```

```
  constructor ∃-lens
```

```
  field
```

```
    {C} : Set ℓ
```

```
    iso : S ≃ (C × A)
```

```
fst : {A B : Set} → Lens1 (A × B) A
```

```
fst = ∃-lens swap★equiv
```

Lens in Agda 2

where

record **isqinv** { $\ell \ell'$ } { $A : \text{Set } \ell$ } { $B : \text{Set } \ell'$ } ($f : A \rightarrow B$) :

Set ($\ell \sqcup \ell'$) **where**

constructor **qinv**

field

$g : B \rightarrow A$

$\alpha : (f \circ g) \sim \text{id}$

$\beta : (g \circ f) \sim \text{id}$

$\underline{\simeq} : \forall \{ \ell \ell' \} \rightarrow \text{Set } \ell \rightarrow \text{Set } \ell' \rightarrow \text{Set } (\ell \sqcup \ell')$

$A \simeq B = \sum (A \rightarrow B) \text{ isqinv}$

Lens in Agda 2

Or, the return of constant-complement lenses:

```
record Lens1 {ℓ : Level} (S : Set ℓ) (A : Set ℓ) : Set (suc ℓ) where
  constructor ∃-lens
  field
    {C} : Set ℓ
    iso : S ≃ (C × A)
```

Lens in Agda 2

Or, the return of constant-complement lenses:

```
record Lens1 {ℓ : Level} (S : Set ℓ) (A : Set ℓ) : Set (suc ℓ) where
  constructor ∃-lens
  field
    {C} : Set ℓ
    iso : S ≃ (C × A)
```

```
sound : {ℓ : Level} {S A : Set ℓ} → Lens1 S A → GS-Lens S A
```

Lens in Agda 2

Or, the return of constant-complement lenses:

```
record Lens1 {ℓ : Level} (S : Set ℓ) (A : Set ℓ) : Set (suc ℓ) where
  constructor ∃-lens
  field
    {C} : Set ℓ
    iso : S ≃ (C × A)
```

`sound` : {ℓ : Level} {S A : Set ℓ} → Lens₁ S A → GS-Lens S A

`complete` requires moving to `Setoid` – see online code.

```
__≈__under__ : ∀ {ℓ} {S A : Set ℓ} → (s t : S) (l : GS-Lens S A) → Set ℓ
__≈__under__ s t l = ∀ a → set l s a ≡ set l t a
```

Exploiting type equivalences

module `_` {`A B D` : `Set`} where

`l1` : `Lens1` `A` `A`

`l1` = `∃-lens` `uniti★equiv`

`l2` : `Lens1` (`B` × `A`) `A`

`l2` = `∃-lens` `id`≃

`l3` : `Lens1` (`B` × `A`) `B`

`l3` = `∃-lens` `swap★equiv`

`l4` : `Lens1` (`D` × (`B` × `A`)) `A`

`l4` = `∃-lens` `assocl★equiv`

`l5` : `Lens1` ⊥ `A`

`l5` = `∃-lens` `factorzequiv`

`l6` : `Lens1` ((`D` × `A`) ⊔ (`B` × `A`)) `A`

`l6` = `∃-lens` `factorequiv`

`uniti★equiv` : $A \simeq (\top \times A)$

`id`≃ : $A \simeq A$

`swap★equiv` : $A \times B \simeq B \times A$

`assocl★equiv` : $(A \times B) \times C \simeq A \times (B \times C)$

`factorzequiv` : $\perp \simeq (\perp \times A)$

`factorequiv` : $((A \times D) \uplus (B \times D)) \simeq ((A \uplus B) \times D)$

Proof relevance

Different proofs that $A \times A \simeq A \times A$ give different lenses:

$l_7 : \text{Lens}_1 (A \times A) A$

$l_7 = \exists\text{-lens id} \simeq$

$l_8 : \text{Lens}_1 (A \times A) A$

$l_8 = \exists\text{-lens swap} \star \text{equiv}$

Plain Curry-Howard: $A \wedge A \equiv A$ implies $A \times A \longleftrightarrow A$ (equi-inhabitation).

Type Equivalences

Semiring

$$a = a$$

$$0 + a = a$$

$$a + b = b + a$$

$$a + (b + c) = (a + b) + c$$

$$1 \cdot a = a$$

$$a \cdot b = b \cdot a$$

$$a \cdot (b \cdot c) = (a \cdot b) \cdot c$$

$$0 \cdot a = 0$$

$$(a + b) \cdot c = (a \cdot c) + (b \cdot c)$$

Type Equivalences

Semiring

$$a = a$$

$$0 + a = a$$

$$a + b = b + a$$

$$a + (b + c) = (a + b) + c$$

$$1 \cdot a = a$$

$$a \cdot b = b \cdot a$$

$$a \cdot (b \cdot c) = (a \cdot b) \cdot c$$

$$0 \cdot a = 0$$

$$(a + b) \cdot c = (a \cdot c) + (b \cdot c)$$

Types

$$A \simeq A$$

$$\perp \uplus A \simeq A$$

$$A \uplus B \simeq B \uplus A$$

$$A \uplus (B \uplus C) \simeq (A \uplus B) \uplus C$$

$$\top * A \simeq A$$

$$A * B \simeq B * A$$

$$A * (B * C) \simeq (A * B) * C$$

$$\perp * A \simeq \perp$$

$$(A \uplus B) * C \simeq (A * C) \uplus (B * C)$$

Categorify: weak symmetric Rig Groupoids!

Let $c_1 : t_1 \leftrightarrow t_2$, $c_2 : t_3 \leftrightarrow t_4$, $c_3 : t_1 \leftrightarrow t_2$, and $c_4 : t_3 \leftrightarrow t_4$.

Let $a_1 : t_5 \leftrightarrow t_1$, $a_2 : t_6 \leftrightarrow t_2$, $a_3 : t_1 \leftrightarrow t_3$, and $a_4 : t_2 \leftrightarrow t_4$.

$$\begin{array}{ccc} \frac{c_1 \leftrightarrow c_3 \quad c_2 \leftrightarrow c_4}{c_1 \oplus c_2 \leftrightarrow c_3 \oplus c_4} & \frac{c_1 \leftrightarrow c_3 \quad c_2 \leftrightarrow c_4}{c_1 \otimes c_2 \leftrightarrow c_3 \otimes c_4} & \\ (a_1 \odot a_3) \oplus (a_2 \odot a_4) \leftrightarrow (a_1 \oplus a_2) \odot (a_3 \oplus a_4) & & \\ (a_1 \odot a_3) \otimes (a_2 \odot a_4) \leftrightarrow (a_1 \otimes a_2) \odot (a_3 \otimes a_4) & & \end{array}$$

Categorify: weak symmetric Rig Groupoids!

Let $c_1 : t_1 \leftrightarrow t_2$, $c_2 : t_2 \leftrightarrow t_3$, and $c_3 : t_3 \leftrightarrow t_4$:

$$c_1 \odot (c_2 \odot c_3) \Leftrightarrow (c_1 \odot c_2) \odot c_3$$

$$(c_1 \oplus (c_2 \oplus c_3)) \odot \text{assocl}_+ \Leftrightarrow \text{assocl}_+ \odot ((c_1 \oplus c_2) \oplus c_3)$$

$$(c_1 \otimes (c_2 \otimes c_3)) \odot \text{assocl}_\times \Leftrightarrow \text{assocl}_\times \odot ((c_1 \otimes c_2) \otimes c_3)$$

$$((c_1 \oplus c_2) \oplus c_3) \odot \text{assocr}_+ \Leftrightarrow \text{assocr}_+ \odot (c_1 \oplus (c_2 \oplus c_3))$$

$$((c_1 \otimes c_2) \otimes c_3) \odot \text{assocr}_\times \Leftrightarrow \text{assocr}_\times \odot (c_1 \otimes (c_2 \otimes c_3))$$

$$\text{assocr}_+ \odot \text{assocr}_+ \Leftrightarrow ((\text{assocr}_+ \oplus \text{id}\leftrightarrow) \odot \text{assocr}_+) \odot (\text{id}\leftrightarrow \oplus \text{assocr}_+)$$

$$\text{assocr}_\times \odot \text{assocr}_\times \Leftrightarrow ((\text{assocr}_\times \otimes \text{id}\leftrightarrow) \odot \text{assocr}_\times) \odot (\text{id}\leftrightarrow \otimes \text{assocr}_\times)$$

Categorify: weak symmetric Rig Groupoids!

Let $c_1 : t_1 \leftrightarrow t_2$, $c_2 : t_3 \leftrightarrow t_4$, and $c_3 : t_5 \leftrightarrow t_6$:

$$((c_1 \oplus c_2) \otimes c_3) \odot \text{dist} \Leftrightarrow \text{dist} \odot ((c_1 \otimes c_3) \oplus (c_2 \otimes c_3))$$

$$(c_1 \otimes (c_2 \oplus c_3)) \odot \text{distl} \Leftrightarrow \text{distl} \odot ((c_1 \otimes c_2) \oplus (c_1 \otimes c_3))$$

$$((c_1 \otimes c_3) \oplus (c_2 \otimes c_3)) \odot \text{factor} \Leftrightarrow \text{factor} \odot ((c_1 \oplus c_2) \otimes c_3)$$

$$((c_1 \otimes c_2) \oplus (c_1 \otimes c_3)) \odot \text{factorl} \Leftrightarrow \text{factorl} \odot (c_1 \otimes (c_2 \oplus c_3))$$

Categorify: weak symmetric Rig Groupoids!

Let $c_0, c_1, c_2, c_3 : t_1 \leftrightarrow t_2$ and $c_4, c_5 : t_3 \leftrightarrow t_4$:

$$id \leftrightarrow \odot c_0 \Leftrightarrow c_0 \quad c_0 \odot id \leftrightarrow \Leftrightarrow c_0 \quad c_0 \odot ! c_0 \Leftrightarrow id \leftrightarrow \quad ! c_0 \odot c_0 \Leftrightarrow id \leftrightarrow$$

$$id \leftrightarrow \oplus id \leftrightarrow \Leftrightarrow id \leftrightarrow \quad id \leftrightarrow \otimes id \leftrightarrow \Leftrightarrow id \leftrightarrow$$

$$c_0 \Leftrightarrow c_0 \quad \frac{c_1 \Leftrightarrow c_2 \quad c_2 \Leftrightarrow c_3}{c_1 \Leftrightarrow c_3} \quad \frac{c_1 \Leftrightarrow c_4 \quad c_2 \Leftrightarrow c_5}{c_1 \odot c_2 \Leftrightarrow c_4 \odot c_5}$$

Categorify: weak symmetric Rig Groupoids!

Let $c_0 : 0 \leftrightarrow 0$, $c_1 : 1 \leftrightarrow 1$, and $c_3 : t_1 \leftrightarrow t_2$:

$$\begin{aligned} unite_+ l \odot c_3 &\Leftrightarrow (c_0 \oplus c_3) \odot unite_+ l & unit_+ l \odot (c_0 \oplus c_3) &\Leftrightarrow c_3 \odot unit_+ l \\ unite_+ r \odot c_3 &\Leftrightarrow (c_3 \oplus c_0) \odot unite_+ r & unit_+ r \odot (c_3 \oplus c_0) &\Leftrightarrow c_3 \odot unit_+ r \\ unite_{\times} l \odot c_3 &\Leftrightarrow (c_1 \otimes c_3) \odot unite_{\times} l & unit_{\times} l \odot (c_1 \otimes c_3) &\Leftrightarrow c_3 \odot unit_{\times} l \\ unite_{\times} r \odot c_3 &\Leftrightarrow (c_3 \otimes c_1) \odot unite_{\times} r & unit_{\times} r \odot (c_3 \otimes c_1) &\Leftrightarrow c_3 \odot unit_{\times} r \\ unite_{\times} l &\Leftrightarrow distl \odot (unite_{\times} l \oplus unite_{\times} l) \\ unite_+ l &\Leftrightarrow swap_+ \odot unite_+ r & unite_{\times} l &\Leftrightarrow swap_{\times} \odot unite_{\times} r \end{aligned}$$

Categorify: weak symmetric Rig Groupoids!

Let $c_1 : t_1 \leftrightarrow t_2$ and $c_2 : t_3 \leftrightarrow t_4$:

$$\begin{aligned} \text{swap}_+ \odot (c_1 \oplus c_2) &\Leftrightarrow (c_2 \oplus c_1) \odot \text{swap}_+ & \text{swap}_\times \odot (c_1 \otimes c_2) &\Leftrightarrow (c_2 \otimes c_1) \odot \text{swap}_\times \\ (\text{assocr}_+ \odot \text{swap}_+) \odot \text{assocr}_+ &\Leftrightarrow ((\text{swap}_+ \oplus \text{id}\leftrightarrow) \odot \text{assocr}_+) \odot (\text{id}\leftrightarrow \oplus \text{swap}_+) \\ (\text{assocl}_+ \odot \text{swap}_+) \odot \text{assocl}_+ &\Leftrightarrow ((\text{id}\leftrightarrow \oplus \text{swap}_+) \odot \text{assocl}_+) \odot (\text{swap}_+ \oplus \text{id}\leftrightarrow) \\ (\text{assocr}_\times \odot \text{swap}_\times) \odot \text{assocr}_\times &\Leftrightarrow ((\text{swap}_\times \otimes \text{id}\leftrightarrow) \odot \text{assocr}_\times) \odot (\text{id}\leftrightarrow \otimes \text{swap}_\times) \\ (\text{assocl}_\times \odot \text{swap}_\times) \odot \text{assocl}_\times &\Leftrightarrow ((\text{id}\leftrightarrow \otimes \text{swap}_\times) \odot \text{assocl}_\times) \odot (\text{swap}_\times \otimes \text{id}\leftrightarrow) \end{aligned}$$

$$\text{unite}_+ r \oplus \text{id}\leftrightarrow \Leftrightarrow \text{assocr}_+ \odot (\text{id}\leftrightarrow \oplus \text{unite}_+ l)$$

$$\text{unite}_\times r \otimes \text{id}\leftrightarrow \Leftrightarrow \text{assocr}_\times \odot (\text{id}\leftrightarrow \otimes \text{unite}_\times l)$$

Categorify: weak symmetric Rig Groupoids!

Let $c : t_1 \leftrightarrow t_2$:

$$\begin{aligned}(c \otimes id \leftrightarrow) \odot absorbl &\Leftrightarrow absorbl \odot id \leftrightarrow & (id \leftrightarrow \otimes c) \odot absorbr &\Leftrightarrow absorbr \odot id \leftrightarrow \\ id \leftrightarrow \odot factorzl &\Leftrightarrow factorzl \odot (id \leftrightarrow \otimes c) & id \leftrightarrow \odot factorzr &\Leftrightarrow factorzr \odot (c \otimes absorbr) \\ absorbr &\Leftrightarrow absorbl \\ absorbr &\Leftrightarrow (distl \odot (absorbr \oplus absorbr)) \odot unite_+ l \\ unite_{\times} r &\Leftrightarrow absorbr & absorbl &\Leftrightarrow swap_{\times} \odot absorbr \\ absorbr &\Leftrightarrow (assocl_{\times} \odot (absorbr \otimes id \leftrightarrow)) \odot absorbr \\ (id \leftrightarrow \otimes absorbr) \odot absorbl &\Leftrightarrow (assocl_{\times} \odot (absorbl \otimes id \leftrightarrow)) \odot absorbr \\ id \leftrightarrow \otimes unite_+ l &\Leftrightarrow (distl \odot (absorbl \oplus id \leftrightarrow)) \odot unite_+ l\end{aligned}$$

Categorify: weak symmetric Rig Groupoids!

$$\begin{aligned} ((\text{assocl}_+ \otimes \text{id} \leftrightarrow) \odot \text{dist}) \odot (\text{dist} \oplus \text{id} \leftrightarrow) &\Leftrightarrow (\text{dist} \odot (\text{id} \leftrightarrow \oplus \text{dist})) \odot \text{assocl}_+ \\ \text{assocl}_\times \odot \text{distl} &\Leftrightarrow ((\text{id} \leftrightarrow \otimes \text{distl}) \odot \text{distl}) \odot (\text{assocl}_\times \oplus \text{assocl}_\times) \end{aligned}$$

$$\begin{aligned} (\text{distl} \odot (\text{dist} \oplus \text{dist})) \odot \text{assocl}_+ &\Leftrightarrow \text{dist} \odot (\text{distl} \oplus \text{distl}) \odot \text{assocl}_+ \odot \\ &(\text{assocr}_+ \oplus \text{id} \leftrightarrow) \odot \\ &((\text{id} \leftrightarrow \oplus \text{swap}_+) \oplus \text{id} \leftrightarrow) \odot \\ &(\text{assocl}_+ \oplus \text{id} \leftrightarrow) \end{aligned}$$

$$\begin{aligned} (\text{id} \leftrightarrow \otimes \text{swap}_+) \odot \text{distl} &\Leftrightarrow \text{distl} \odot \text{swap}_+ \\ \text{dist} \odot (\text{swap}_\times \oplus \text{swap}_\times) &\Leftrightarrow \text{swap}_\times \odot \text{distl} \end{aligned}$$

So what?

Up shot: coherence theorem (like the one for monoidal categories).

Sound and complete set of rewrites / optimization rules for type equivalences.

More lens programs

`data Colour : Set where red green blue : Colour`

`∃-Colour-in-A+A+A : (A : Set) → Lens1 (A ⊔ A ⊔ A) Colour`

More lens programs

`data Colour : Set where red green blue : Colour`

`∃-Colour-in-A+A+A : (A : Set) → Lens1 (A ⊔ A ⊔ A) Colour`

`GS-Colour-in-A+A+A : (A : Set) → GS-Lens (A ⊔ A ⊔ A) Colour`

For $n \in \mathbb{N}$ and nA , `Lens1` proof terms $O(n)$, `GS-Lens` are $O(n^3)$.

More lens programs

`data Colour : Set where red green blue : Colour`

`∃-Colour-in-A+A+A : (A : Set) → Lens1 (A ⊔ A ⊔ A) Colour`

`GS-Colour-in-A+A+A : (A : Set) → GS-Lens (A ⊔ A ⊔ A) Colour`

For $n \in \mathbb{N}$ and nA , `Lens1` proof terms $O(n)$, `GS-Lens` are $O(n^3)$.

Inspired by Quantum Computing:

`gcnot-equiv : {A B C : Set} →`

`((A ⊔ B) × (C ⊔ C)) ≃ ((A ⊔ B) × (C ⊔ C))`

`gcnot-equiv = factorequiv • id≃ ⊔≃ (id≃ ×≃ swap+equiv) • distequiv`

`gcnot-lens : {A B C : Set} → Lens1 ((A ⊔ B) × (C ⊔ C)) (C ⊔ C)`

`gcnot-lens {A} {B} = ∃-lens gcnot-equiv`

Equality	$S = A$
Iso	$S \simeq A$
Lens	$\exists C. S \simeq C \times A$
Prism	$\exists C. S \simeq C + A$
Achroma	$\exists C. S \simeq (\top \uplus C) \times A$
Affine	$\exists C, D. S \simeq D \uplus (C \times A)$
Grate	$\exists I. S \simeq I \rightarrow A$
Setter	$\exists F : Set \rightarrow Set. S \simeq FA$

Geometric interpretation

Iso	$S \simeq A$	S is A up to change of coordinates (CoC)
Lens	$\exists C. S \simeq C \times A$	S is a <i>cartesian product</i> over A , up to CoC
Prism	$\exists C. S \simeq C + A$	S has a <i>partition</i> into A and C , up to CoC
