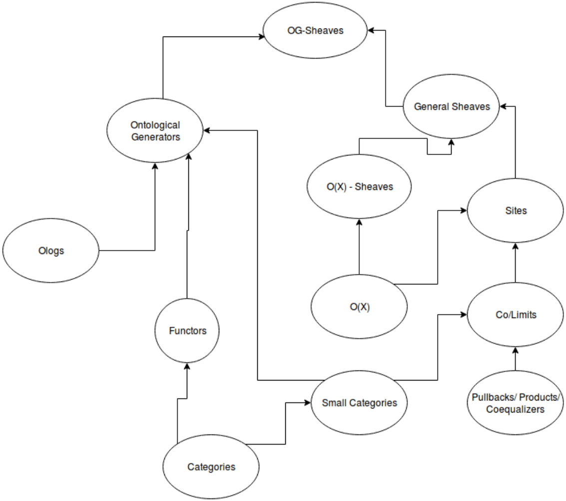
Categories, Sheaves; Applications, Ologs

Noah Chrein

March 4, 2019

Content Ontology



References

[CWM] Categories for the Working Mathematician - Mac Lane
[SGL] Sheaves in Geometry and Logic - Mac Lane
[STACK] Sites - Stacks Project
[LEARN] Backprop as Functor - Brandon Fong, David Spivak
[SSA] Sheaves, Cosheaves and Applications - Justin Curry
[OLOG] Ontological Logs - David Spivak

Categories

Def : Category

A category $\mathfrak{C}$ is a collection of objects $(A, B, X, Y, ... \in Ob(\mathfrak{C}))$ and morphisms $(f : A \to B)$ such that:

Categories

Def : Category

A category $\mathfrak{C}$ is a collection of objects $(A, B, X, Y, ... \in Ob(\mathfrak{C}))$ and morphisms $(f : A \to B)$ such that:

- The collection of morphisms $f : A \to B$ form a **Set** $Hom_{\mathfrak{C}}(A, B)$

Categories

Def : Category

A category $\mathfrak{C}$ is a collection of objects $(A, B, X, Y, ... \in Ob(\mathfrak{C}))$ and morphisms $(f : A \to B)$ such that:

- The collection of morphisms $f : A \to B$ form a **Set** $Hom_{\mathfrak{C}}(A, B)$
- There is an associative and unital law of composition $\circ : Hom_{\mathfrak{C}}(B, C) \times Hom_{\mathfrak{C}}(A, B) \to Hom_{\mathfrak{C}}(A, C)$

Categories

Def : Category

A category $\mathfrak{C}$ is a collection of objects $(A, B, X, Y, ... \in Ob(\mathfrak{C}))$ and morphisms $(f : A \to B)$ such that:

- The collection of morphisms $f : A \to B$ form a **Set** $Hom_{\mathfrak{C}}(A, B)$
- There is an associative and unital law of composition $\circ : Hom_{\mathfrak{C}}(B, C) \times Hom_{\mathfrak{C}}(A, B) \to Hom_{\mathfrak{C}}(A, C)$

we typically write the composition of two morphisms by $f \circ g$.

Categories

Def : Category

A category $\mathfrak{C}$ is a collection of objects $(A, B, X, Y, ... \in Ob(\mathfrak{C}))$ and morphisms $(f : A \to B)$ such that:

- The collection of morphisms $f : A \to B$ form a **Set** $Hom_{\mathfrak{C}}(A, B)$
- There is an associative and unital law of composition $\circ : Hom_{\mathfrak{C}}(B, C) \times Hom_{\mathfrak{C}}(A, B) \to Hom_{\mathfrak{C}}(A, C)$

we typically write the composition of two morphisms by $f \circ g$.
Associativity implies $(f \circ g) \circ h = f \circ (g \circ h)$.
Unital is the existence of "identity morphisms" $Id_C \in Hom(C, C)$ with $Id_B \circ f = f = f \circ Id_A$

Examples of Categories

Some examples include $\mathfrak{S}et$, $\mathfrak{T}op$, $\mathfrak{G}rp$, $\mathfrak{A}b$, $\mathfrak{R}ing$, $\mathfrak{M}od_R$

Examples of Categories

Some examples include $\mathfrak{S}et$, $\mathfrak{T}op$, $\mathfrak{G}rp$, $\mathfrak{A}b$, $\mathfrak{R}ing$, $\mathfrak{M}od_R$

- Categories capture the idea of structure and structure preserving relations.

Examples of Categories

Some examples include $\mathfrak{S}et$, $\mathfrak{T}op$, $\mathfrak{G}rp$, $\mathfrak{A}b$, $\mathfrak{R}ing$, $\mathfrak{M}od_R$

- Categories capture the idea of structure and structure preserving relations.
- In general, individual objects of a category need not be sets, the morphisms need not be functions, and the collection of objects $Ob(\mathfrak{C})$ need not form a set.

Examples of Categories

Some examples include $\mathfrak{Set}$, $\mathfrak{Top}$, $\mathfrak{Grp}$, $\mathfrak{Ab}$, $\mathfrak{Ring}$, $\mathfrak{Mod}_R$

- Categories capture the idea of structure and structure preserving relations.
- In general, individual objects of a category need not be sets, the morphisms need not be functions, and the collection of objects $Ob(\mathfrak{C})$ need not form a set.

Def : Small Category

A **Small Category** is a category in which the objects form a set.

Examples: $(\mathbb{R}, \leq)$, $O(X)$

O(X)

Def : O(X)

Let X be a topological space. **O(X)** is the small category whose objects are the open sets of X, and whose morphisms are the inclusions $i_{U,V} : U \hookrightarrow V$ (that is when $U \subseteq V$)

O(X)

Def : O(X)

Let X be a topological space. **O(X)** is the small category whose objects are the open sets of X, and whose morphisms are the inclusions $i_{U,V} : U \hookrightarrow V$ (that is when $U \subseteq V$)

- Self inclusion gives the identity map, and composition is given by transitivity of $\subseteq$

O(X)

Def : O(X)

Let X be a topological space. **O(X)** is the small category whose objects are the open sets of X, and whose morphisms are the inclusions $i_{U,V} : U \hookrightarrow V$ (that is when $U \subseteq V$)

- Self inclusion gives the identity map, and composition is given by transitivity of $\subseteq$
- For every $U \in O(x)$ we can consider an **open covering** of U, $\{U_i \to U\}$ (that is, $\bigcup U_i = U$)
- For two $U, V \in O(X)$ we can consider the intersection $U \cap V \in O(X)$

Functors

We want a way to retrieve relevant data from objects in our category

Functors

We want a way to retrieve relevant data from objects in our category

Def : Functors

Let $\mathfrak{C}$, $\mathfrak{D}$ be two categories. A **functor** $F : \mathfrak{C} \to \mathfrak{D}$ is a rule $F : ob(\mathfrak{C}) \to ob(\mathfrak{D})$ and $F : mor(\mathfrak{C}) \to mor(\mathfrak{D})$ such that:

- if $f : A \to B$ then $F(f) : F(A) \to F(B)$
- $F(f \circ g) = F(f) \circ F(g)$
- $F(Id_C) = Id_{F(C)}$

Functors

We want a way to retrieve relevant data from objects in our category

Def : Functors

Let $\mathfrak{C}$, $\mathfrak{D}$ be two categories. A **functor** $F : \mathfrak{C} \to \mathfrak{D}$ is a rule $F : ob(\mathfrak{C}) \to ob(\mathfrak{D})$ and $F : mor(\mathfrak{C}) \to mor(\mathfrak{D})$ such that:

- if $f : A \to B$ then $F(f) : F(A) \to F(B)$
- $F(f \circ g) = F(f) \circ F(g)$
- $F(Id_C) = Id_{F(C)}$

There is also the notion of a **contravariant functor**, a functor such that $F(f : A \to B) = F(f) : F(B) \to F(A)$, in this case we write $F : \mathfrak{C}^{op} \to \mathfrak{D}$

Examples / Intuition

- relevant examples include $\pi_1 : \mathfrak{T}op \to \mathfrak{G}rp$, $H_n : \mathfrak{T}op \to \mathfrak{A}b$, $H^n : \mathfrak{T}op^{op} \to \mathfrak{A}b$, $C(-, Y) : \mathfrak{T}op^{op} \to \mathfrak{S}et$, $O : \mathfrak{T}op \to Sm(Top)$

Examples / Intuition

- relevant examples include $\pi_1 : \mathfrak{T}op \to \mathfrak{G}rp$, $H_n : \mathfrak{T}op \to \mathfrak{A}b$, $H^n : \mathfrak{T}op^{op} \to \mathfrak{A}b$, $C(-, Y) : \mathfrak{T}op^{op} \to \mathfrak{S}et$, $O : \mathfrak{T}op \to Sm(Top)$
- $C(X, Y) = \{f : X \to Y | \text{continuous}\}$, if $f : A \to X$ then $C(f, Y) = f^* : C(X, Y) \to C(A, Y)$ by $\phi \mapsto \phi \circ f$

The **intuition** behind a functor is that it captures $\mathfrak{C}$ - invariants valued in $\mathfrak{D}$

Examples / Intuition

- relevant examples include $\pi_1 : \mathfrak{T}op \to \mathfrak{G}rp$, $H_n : \mathfrak{T}op \to \mathfrak{A}b$, $H^n : \mathfrak{T}op^{op} \to \mathfrak{A}b$, $C(-, Y) : \mathfrak{T}op^{op} \to \mathfrak{S}et$, $O : \mathfrak{T}op \to Sm(Top)$
- $C(X, Y) = \{f : X \to Y | \text{continuous}\}$, if $f : A \to X$ then $C(f, Y) = f^* : C(X, Y) \to C(A, Y)$ by $\phi \mapsto \phi \circ f$

The **intuition** behind a functor is that it captures $\mathfrak{C}$ - invariants valued in $\mathfrak{D}$

This is great, but a functor tells us the "global" invariants of a space, we would like to find local ones.

Presheaves, C(-,Y)

This notion of "locality" is captured by a presheaf:

Presheaves, C(-,Y)

This notion of "locality" is captured by a presheaf:

Def : Presheaf

a **presheaf** on a space X is a contravariant functor $F : O(X)^{op} \to \mathfrak{S}et$

Presheaves, C(-,Y)

This notion of "locality" is captured by a presheaf:

Def : Presheaf

a **presheaf** on a space X is a contravariant functor
$F : O(X)^{op} \to \mathfrak{S}et$

An example of a presheaf is $C(-, Y)$. In this case:
$C(U, Y) = \{f : U \to Y | \text{continuous}\}$
$C(i_{U,V}, Y)(f : V \to Y) = f \circ i_{U,V} = f|_U : U \to Y.$

Properties of $C(-,Y) : O(X)^{op} \to \mathfrak{S}et$

$C(-,Y)$ enjoys some nice properties:

Given $\{U_i \to U\}$ an (open) covering:
1)**gluing**: if $f_i : U_i \to Y$ such that

$$f_i|_{U_i \cap U_j} = f_j|_{U_i \cap U_j}$$

then $\exists! f : U \to Y$ such that $f|_{U_i} = f_i$
2)**Locality**: If $f, g : U \to Y$ such that

$$f|_{U_i} = g|_{U_i}$$

then $f = g$

In this case we say that the presheaf $C(-,Y)$ is a **sheaf**

Properties of $C(-,Y) : O(X)^{op} \to \mathfrak{S}et$

$C(-,Y)$ enjoys some nice properties:

Given $\{U_i \to U\}$ an (open) covering:
1)**gluing**: if $f_i \in C(U_i, Y)$ such that

$$f_i|_{U_i \cap U_j} = f_j|_{U_i \cap U_j}$$

then $\exists! f : U \to Y$ such that $f|_{U_i} = f_i$
2)**Locality**: If $f, g : U \to Y$ such that

$$f|_{U_i} = g|_{U_i}$$

then $f = g$

In this case we say that the presheaf $C(-,Y)$ is a **sheaf**

Properties of $C(-, Y) : O(X)^{op} \to \mathfrak{S}et$

$C(-, Y)$ enjoys some nice properties:

Given $\{U_i \to U\}$ an (open) covering:
1)**gluing**: if $f_i \in C(U_i, Y)$ such that

$$C(i_{U_i \cap U_j, U_i}, Y)(f_i) = C(i_{U_i \cap U_j, U_j}, Y)(f_j)$$

then $\exists! f : U \to Y$ such that $f|_{U_i} = f_i$
2)**Locality**: If $f, g : U \to Y$ such that

$$f|_{U_i} = g|_{U_i}$$

then $f = g$

In this case we say that the presheaf $C(-, Y)$ is a **sheaf**

Properties of $C(-,Y) : O(X)^{op} \to \mathfrak{S}et$

$C(-,Y)$ enjoys some nice properties:

Given $\{U_i \to U\}$ an (open) covering:
1)**gluing**: if $f_i \in C(U_i, Y)$ such that

$$C(i_{U_i \cap U_j, U_i}, Y)(f_i) = C(i_{U_i \cap U_j, U_j}, Y)(f_j)$$

then $\exists ! f \in C(U, Y)$ such that $C(i_{U_i, U}, Y)(f) = f_i$
2)**Locality**: If $f, g : U \to Y$ such that

$$f|_{U_i} = g|_{U_i}$$

then $f = g$

In this case we say that the presheaf $C(-,Y)$ is a **sheaf**

Properties of $C(-, Y) : O(X)^{op} \to \mathfrak{S}et$

$C(-, Y)$ enjoys some nice properties:

Given $\{U_i \to U\}$ an (open) covering:
1)**gluing**: if $f_i \in C(U_i, Y)$ such that

$$C(i_{U_i \cap U_j, U_i}, Y)(f_i) = C(i_{U_i \cap U_j, U_j}, Y)(f_j)$$

then $\exists ! f \in C(U, Y)$ such that $C(i_{U_i, U}, Y)(f) = f_i$
2)**Locality**: If $f, g \in C(U, Y)$ such that

$$C(i_{U_i, U}, Y)(f) = C(i_{U_i, U}, Y)(g)$$

then $f = g$

In this case we say that the presheaf $C(-, Y)$ is a **sheaf**

Sheaves

Def : Sheaf

A presheaf $P : O(X)^{op} \to Y$ is a **sheaf** if,
given $\{U_i \to U\}$ an (open) covering:
1)**gluing**: if $f_i \in P(U_i)$ such that

$$P(i_{U_i \cap U_j, U_i})(f_i) = P(i_{U_i \cap U_j, U_j})(f_j)$$

then $\exists ! f \in P(U)$ such that $P(i_{U_i,U})(f) = f_i$
2)**Locality**: If $f, g \in P(U)$ such that

$$P(i_{U_i,U})(f) = P(i_{U_i,U})(g)$$

then $f = g$

Intuition

-A **category** captures the idea of mathematical structure and structure preserving relations
-A **functor** extracts data from objects with structure
-A **presheaf** extracts local data from an object
-A **sheaf** extracts local data from an object that can be used to build global data.

Pit Stop for Applications

- Brandon Fong and his advisor David Spivak recently described Backpropogation in reinforncement learning as a functor (good because it captures compositionality of learning) [LEARN]
- Michael Sent me a giant paper with tons of applications, one of which I thought was very cool: Viewing the coverage area of a bunch of cameras and the data they retrieve in terms of sheaves. [SSA]

Generalization Scheme

In order to talk about the application to OLOGs, we must remove our set theoretic crutches.

Generalization Scheme

In order to talk about the application to OLOGs, we must remove our set theoretic crutches.

$P : O(X)^{op} \to \mathfrak{Set} \implies P : \mathfrak{C}^{op} \to \mathfrak{D}$

Generalization Scheme

In order to talk about the application to OLOGs, we must remove our set theoretic crutches.

$P : O(X)^{op} \to \mathfrak{Set} \implies P : \mathfrak{C}^{op} \to \mathfrak{D}$

- $O(X) \implies \mathfrak{C}$ "Site"

Generalization Scheme

In order to talk about the application to OLOGs, we must remove our set theoretic crutches.

$P : O(X)^{op} \to \mathfrak{Set} \implies P : \mathfrak{C}^{op} \to \mathfrak{D}$

- $O(X) \implies \mathfrak{C}$ "Site"
  $U \cap V \implies U \times_X V$ "Pullback"

Generalization Scheme

In order to talk about the application to OLOGs, we must remove our set theoretic crutches.

$P : O(X)^{op} \to \mathfrak{Set} \implies P : \mathfrak{C}^{op} \to \mathfrak{D}$

- $O(X) \implies \mathfrak{C}$ "Site"
  $U \cap V \implies U \times_X V$ "Pullback"
  $\bigcup U_i = U \implies \{U_i \to U\}$ "Covering"

Generalization Scheme

In order to talk about the application to OLOGs, we must remove our set theoretic crutches.

$P : O(X)^{op} \to \mathfrak{Set} \implies P : \mathfrak{C}^{op} \to \mathfrak{D}$

- $O(X) \implies \mathfrak{C}$ "Site"
  $U \cap V \implies U \times_X V$ "Pullback"
  $\bigcup U_i = U \implies \{U_i \to U\}$ "Covering"
- $\mathfrak{Set} \implies \mathfrak{D}$ "Complete Category"

Generalization Scheme

In order to talk about the application to OLOGs, we must remove our set theoretic crutches.

$P : O(X)^{op} \to \mathfrak{Set} \implies P : \mathfrak{C}^{op} \to \mathfrak{D}$

- $O(X) \implies \mathfrak{C}$ "Site"
  $U \cap V \implies U \times_X V$ "Pullback"
  $\bigcup U_i = U \implies \{U_i \to U\}$ "Covering"
- $\mathfrak{Set} \implies \mathfrak{D}$ "Complete Category"
  $\{f_i \in P(U_i)\}_{i \in I} \implies \prod_{i \in I} P(U_i)$ "Product"

Generalization Scheme

In order to talk about the application to OLOGs, we must remove our set theoretic crutches.

$P : O(X)^{op} \to \mathfrak{Set} \implies P : \mathfrak{C}^{op} \to \mathfrak{D}$

- $O(X) \implies \mathfrak{C}$ "Site"
  $U \cap V \implies U \times_X V$ "Pullback"
  $\bigcup U_i = U \implies \{U_i \to U\}$ "Covering"
- $\mathfrak{Set} \implies \mathfrak{D}$ "Complete Category"
  $\{f_i \in P(U_i)\}_{i \in I} \implies \prod_{i \in I} P(U_i)$ "Product"
  $f = g \implies Eq(f, g)$ "Equalizer"

Pullbacks, Products and Equalizers (oh my!)

$$\begin{tikzcd}
 & X & \\
U \arrow[ur, "i"] & & V \arrow[ul, "j"']
\end{tikzcd}$$

Given this diagram

Pullbacks, Products and Equalizers (oh my!)

$$\begin{tikzcd}
 & X & \\
U \arrow[ur, "i"] & & V \arrow[ul, "j"'] \\
 & U \times_X V \arrow[ul, "\pi_U"] \arrow[ur, "\pi_V"'] &
\end{tikzcd}$$

Given this diagram, the pullback is an object $U \times_X V$, with two maps π_U, π_V

Pullbacks, Products and Equalizers (oh my!)

$$\begin{tikzcd}
& X & \\
U \arrow[ur, "i"] & & V \arrow[ul, "j"'] \\
& U \times_X V \arrow[ul, "\pi_U"] \arrow[ur, "\pi_V"'] & \\
& Z \arrow[uul, bend left] \arrow[uur, bend right] &
\end{tikzcd}$$

Given this diagram, the pullback is an object $U \times_X V$ completing the diagram with two maps π_U, π_V, such that if any other object Z completes the diagram,

Pullbacks, Products and Equalizers (oh my!)

$$\begin{tikzcd}
 & X & \\
U \arrow[ur, "i"] & & V \arrow[ul, "j"'] \\
 & U \times_X V \arrow[ul, "\pi_U"] \arrow[ur, "\pi_V"'] & \\
 & Z \arrow[uul, bend left] \arrow[uur, bend right] \arrow[u, dashed, "!"] &
\end{tikzcd}$$

Given this diagram, the pullback is an object $U \times_X V$ completing the diagram with two maps π_U, π_V, such that if any other object Z completes the diagram, there is a unique map $Z \to U \times_X V$ making the whole diagram commute

Pullbacks, Products and Equalizers (oh my!)

$$\begin{tikzcd}
 & X & \\
U \arrow[ur, "i"] & & V \arrow[ul, "j"'] \\
 & U \times_X V \arrow[ul, "\pi_U"] \arrow[ur, "\pi_V"'] & \\
 & Z \arrow[u, dashed, "!"] \arrow[uul, bend left] \arrow[uur, bend right] &
\end{tikzcd}$$

- In Set, the pullback is given specifically by $U \times_X V = \{(u, v) \in U \times V | i(u) = j(v)\}$
- Specifically, if U,V are subsets of X and i,j the inclusions, then $U \times_X V = U \cap V$

Pullbacks, Products and Equalizers (oh my!)

$$\begin{tikzcd}
\prod_{i\in I} X_i \arrow[d, "\pi_j"] \\
X_j
\end{tikzcd}$$

Given a collection of objects $\{X_i\}_{i\in I}$ indexed by a set I, we can form the product $\prod_{i\in I} X_i$. This product comes with projection maps π_j

Pullbacks, Products and Equalizers (oh my!)

$$\begin{tikzcd}
\prod_{i\in I} X_i \arrow[d, "\pi_j"] & & Z \arrow[dll] \\
X_j & &
\end{tikzcd}$$

Given a collection of objects $\{X_i\}_{i\in I}$ indexed by a set I, we can form the product $\prod_{i\in I} X_i$. This product comes with projection maps π_j. If any other object comes with maps $Z \to X_i$

Pullbacks, Products and Equalizers (oh my!)

$$\begin{tikzcd}
\prod_{i\in I} X_i \arrow[d, "\pi_j"] & & Z \arrow[ll, dashed, "!"'] \arrow[dll] \\
X_j & &
\end{tikzcd}$$

Given a collection of objects $\{X_i\}_{i\in I}$ indexed by a set I, we can form the product $\prod_{i\in I} X_i$. This product comes with projection maps π_j. If any other object comes with maps $Z \to X_i$, then they must factor through a unique map $Z \to \prod X_i$.

Pullbacks, Products and Equalizers (oh my!)

$$\begin{tikzcd}
\prod_{i\in I} X_i \arrow[d, "\pi_j"] & Z \arrow[l, dashed, "!"] \arrow[dl] \\
X_j &
\end{tikzcd}$$

- Of course the product of sets is the usual cartesian product of set, for spaces it is the cartesian product with the product topology
- Warning! In the same way that not every object is a set, not every product is the cartesian product. (For example in certain cases the product in one can be realized as a pullback in another)

Pullbacks, Products and Equalizers (oh my!)

$$\begin{tikzcd} X \arrow[r, shift left, "f"] \arrow[r, shift right, "g"'] & Y \end{tikzcd}$$

given two maps $f, g : X \to Y$,

Pullbacks, Products and Equalizers (oh my!)

$$\begin{tikzcd}E \arrow[r, "\alpha"] & X \arrow[r, shift left, "f"] \arrow[r, shift right, "g"'] & Y\end{tikzcd}$$

given two maps $f, g : X \to Y$, we can form the equalizer $E = Eq(f, g)$ which comes with a map $\alpha : E \to X$ making $f \circ \alpha = g \circ \alpha$

Pullbacks, Products and Equalizers (oh my!)

$$\begin{tikzcd}
E \arrow[r, "\alpha"] & X \arrow[r, shift left, "f"] \arrow[r, shift right, "g"'] & Y \\
Z \arrow[u, dashed, "!"] \arrow[ur] & &
\end{tikzcd}$$

given two maps $f, g : X \to Y$, we can form the equalizer $E = Eq(f, g)$ which comes with a map $\alpha : E \to X$ making $f \circ \alpha = g \circ \alpha$. If there is a map $Z \to X$ doing the same, then it must factor through the equalizer via a unique map $Z \to E$

Pullbacks, Products and Equalizers (oh my!)

$$\begin{tikzcd}
E \arrow[r, "\alpha"] & X \arrow[r, shift left, "f"] \arrow[r, shift right, "g"'] & Y \\
Z \arrow[u, dashed, "!"] \arrow[ur] & &
\end{tikzcd}$$

- In Set, $E = \{x \in X | f(x) = g(x)\}$
- In Ab, $E = Ker(f - g)$

Generalizing Sheaves

Lets first consider $P = C(-, Y) : O(X)^{op} \to \mathfrak{S}et$.

Generalizing Sheaves

Lets first consider $P = C(-, Y) : O(X)^{op} \to \mathfrak{S}et$.
Let $\{U_i \to U\}$ be an open cover.

Generalizing Sheaves

Lets first consider $P = C(-, Y) : O(X)^{op} \to \mathfrak{S}et$.
Let $\{U_i \to U\}$ be an open cover.
1) for each U_i, the collection $\{U_i \cap U_j \to U_i\}$ is an open covering

Generalizing Sheaves

Lets first consider $P = C(-, Y) : O(X)^{op} \to \mathfrak{S}et$.
Let $\{U_i \to U\}$ be an open cover.
1) for each U_i, the collection $\{U_i \cap U_j \to U_i\}$ is an open covering
2) Consider the functions $res_{U_i, U_i \cap U_j} : P(U_i) \to P(U_i \cap U_j)$ and $res_{U_j, U_i \cap U_j} : P(U_j) \to P(U_i \cap U_j)$, we can lift this to two functions

$$\begin{tikzcd} \prod_i P(U_i) \arrow[r, shift left, "r_1"] \arrow[r, shift right, "r_2"'] & \prod_{i,j} P(U_i \cap U_j) \end{tikzcd}$$

Generalizing Sheaves

Lets first consider $P = C(-, Y) : O(X)^{op} \to \mathfrak{S}et$.
Let $\{U_i \to U\}$ be an open cover.
1) for each U_i, the collection $\{U_i \cap U_j \to U_i\}$ is an open covering
2) Consider the functions $res_{U_i, U_i \cap U_j} : P(U_i) \to P(U_i \cap U_j)$ and $res_{U_j, U_i \cap U_j} : P(U_j) \to P(U_i \cap U_j)$, we can lift this to two functions

$$\begin{tikzcd} \prod_i P(U_i) \arrow[r, shift left, "r_1"] \arrow[r, shift right, "r_2"'] & \prod_{i,j} P(U_i \cap U_j) \end{tikzcd}$$

$r_1(\{f_i : U_i \to Y\}_i) = \{f_i|_{U_i \cap U_j}\}_{i,j}$

Generalizing Sheaves

Lets first consider $P = C(-,Y) : O(X)^{op} \to \mathfrak{S}et$.
Let $\{U_i \to U\}$ be an open cover.
1) for each U_i, the collection $\{U_i \cap U_j \to U_i\}$ is an open covering
2) Consider the functions $res_{U_i, U_i \cap U_j} : P(U_i) \to P(U_i \cap U_j)$ and $res_{U_j, U_i \cap U_j} : P(U_j) \to P(U_i \cap U_j)$, we can lift this to two functions

$$\begin{tikzcd}\prod_i P(U_i) \arrow[r, shift left, "r_1"] \arrow[r, shift right, "r_2"'] & \prod_{i,j} P(U_i \cap U_j)\end{tikzcd}$$

$r_1(\{f_i : U_i \to Y\}_i) = \{f_i|_{U_i \cap U_j}\}_{i,j}$
$r_2(\{f_i : U_i \to Y\}_i) = \{f_i|_{U_j \cap U_i}\}_{i,j}$

Generalizing Sheaves

Lets first consider $P = C(-, Y) : O(X)^{op} \to \mathfrak{Set}$.
Let $\{U_i \to U\}$ be an open cover.
1) for each U_i, the collection $\{U_i \cap U_j \to U_i\}$ is an open covering
2) Consider the functions $res_{U_i, U_i \cap U_j} : P(U_i) \to P(U_i \cap U_j)$ and $res_{U_j, U_i \cap U_j} : P(U_j) \to P(U_i \cap U_j)$, we can lift this to two functions

$$\begin{tikzcd} \prod_i P(U_i) \arrow[r, shift left, "r_1"] \arrow[r, shift right, "r_2"'] & \prod_{i,j} P(U_i \cap U_j) \end{tikzcd}$$

$r_1(\{f_i : U_i \to Y\}_i) = \{f_i|_{U_i \cap U_j}\}_{i,j}$
$r_2(\{f_i : U_i \to Y\}_i) = \{f_i|_{U_j \cap U_i}\}_{i,j}$
4) Finally consider the map $r : P(U) \to \prod_i P(U_i)$
$p(f : U \to Y) = \{f|_{U_i}\}$

General Sheaves

$$\begin{tikzcd} P(U) \arrow[r, "r"] & \prod_i P(U_i) \arrow[r, shift left, "r_1"] \arrow[r, shift right, "r_2"'] & \prod_{i,j} P(U_i \cap U_j) \end{tikzcd}$$

General Sheaves

$$\begin{tikzcd}
P(U) \arrow[r, "r"] & \prod_i P(U_i) \arrow[r, shift left, "r_1"] \arrow[r, shift right, "r_2"'] & \prod_{i,j} P(U_i \cap U_j)
\end{tikzcd}$$

5) Clearly $r_1 \circ r = r_2 \circ r$ as $f|_{U_i}|_{U_i \cap U_j} = f|_{U_i \cap U_j} = f|_{U_j}|_{U_i \cap U_j}$

General Sheaves

$$\begin{tikzcd} P(U) \arrow[r, "r"] & \prod_i P(U_i) \arrow[r, shift left, "r_1"] \arrow[r, shift right, "r_2"'] & \prod_{i,j} P(U_i \cap U_j) \end{tikzcd}$$

5) Clearly $r_1 \circ r = r_2 \circ r$ as $f|_{U_i}|_{U_i \cap U_j} = f|_{U_i \cap U_j} = f|_{U_j}|_{U_i \cap U_j}$
6) if $r(f) = r(g)$ i.e. $f|_{U_i} = g|_{U_i}$ the **locality** sheaf condition implies $f = g$, that is, r is an injection

General Sheaves

$$\begin{tikzcd} P(U) \arrow[r, "r"] & \prod_i P(U_i) \arrow[r, shift left, "r_1"] \arrow[r, shift right, "r_2"'] & \prod_{i,j} P(U_i \cap U_j) \end{tikzcd}$$

5) Clearly $r_1 \circ r = r_2 \circ r$ as $f|_{U_i}|_{U_i \cap U_j} = f|_{U_i \cap U_j} = f|_{U_j}|_{U_i \cap U_j}$
6) if $r(f) = r(g)$ i.e. $f|_{U_i} = g|_{U_i}$ the **locality** sheaf condition implies $f = g$, that is, r is an injection
7)If $\{f_i\}$ is a collection of maps such that $r_1(\{f_i\}) = r_2(\{f_i\})$ then,

$$f_i|_{U_i \cap U_j} = f_j|_{U_i \cap U_j}$$

the **gluing** sheaf condition says that there is a map $f : U \to Y$ such that $r(f) = \{f|_{U_i}\} = \{f_i\}$, i.e. that r is a surjection onto the **equalizer** of r_1, r_2.

General Sheaves

So we can redefine a sheaf on O(X) as a presheaf P such that P(U) is the equalizer of the diagram

$$\begin{tikzcd} P(U) \arrow[r, "r"] & \prod_i P(U_i) \arrow[r, shift left, "r_1"] \arrow[r, shift right, "r_2"'] & \prod_{i,j} P(U_i \cap U_j) \end{tikzcd}$$

Sites

The relevant ideas we used from O(X) is exactly the data of a Site:

Def : Site

A small category $\mathfrak{C}$ is a **site** if it has pullbacks and a collection of coverings $\{U_i \to U\}$ such that:

- if $V \to U$ is an isomorphism, then $\{V \to U\}$ is a covering.
  (An open set covers itself)
- if $\{U_i \to U\}$ a covering and $\{V_{i,j} \to U_i\}$ coverings, then $\{V_{i,j} \to U\}$ is a covering.
  (Refinement of coverings)
- if $\{U_i \to U\}$ is a covering and $V \to U$, then $\{U_i \times_U V \to V\}$ is a covering.
  (if $V \subseteq U$ then $U_i \cap V$ covers V)

General Sheaves

So for a general site $\mathfrak{C}$ and a category $\mathfrak{D}$ with equalizers and products (complete category) we can define a sheaf:

Def : Sheaf

A presheaf $P : \mathfrak{C}^{op} \to \mathfrak{D}$ is a **sheaf** if for every covering $\{U_i \to U\}$, P(U) is the equalizer of the induced sequence

$$\begin{tikzcd} P(U) \arrow[r, "r"] & \prod_i P(U_i) \arrow[r, shift left, "r_1"] \arrow[r, shift right, "r_2"'] & \prod_{i,j} P(U_i \times_U U_j) \end{tikzcd}$$

Quick Recap / Applications

- We defined a sheaf on O(X)

Quick Recap / Applications

- We defined a sheaf on O(X)
- We generalized the notions of "collections, intersections, open coverings, equality"

Quick Recap / Applications

- We defined a sheaf on O(X)
- We generalized the notions of "collections, intersections, open coverings, equality"
- We reformulated sheaf conditions in terms of "products, equalizers and pullbacks"

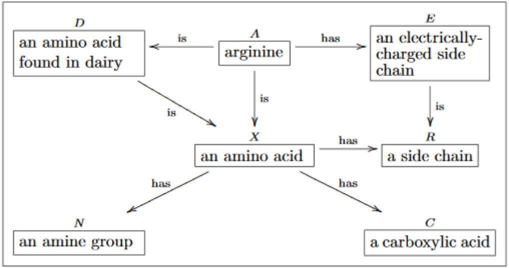
Quick Recap / Applications

- We defined a sheaf on O(X)
- We generalized the notions of "collections, intersections, open coverings, equality"
- We reformulated sheaf conditions in terms of "products, equalizers and pullbacks"
- We lifted the notion of a set-valued sheaf on O(X), to a $\mathfrak{D}$-valued sheaf on a site $\mathfrak{C}$

Besides having applications to geometry, sheaves in this level of generality define a topos, which is a modern tool in logic. All of this can be found in [SGL]

Ontological Logs

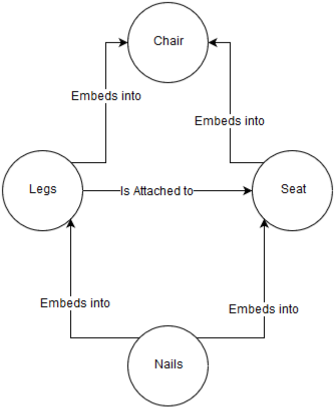
- Concieved by David Spivak (MIT)
- An ontological Log is just a labeled category
- Objects are supposed to capture ideas, and morphisms relations between them



[OLOG]

Ontological Logs

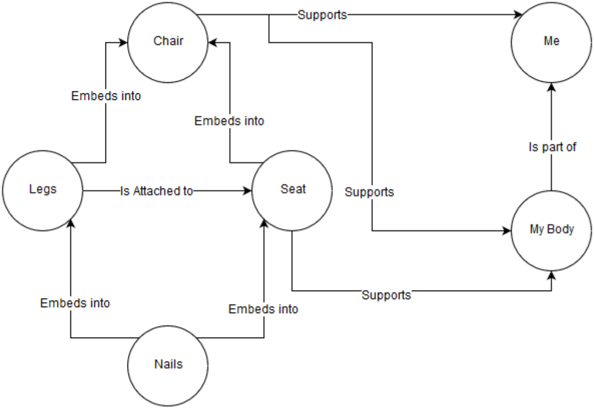
- Categorical constructions can then be interpreted semantically
- Identity ”A concept is itself”
- Composability ”If I effect something, I (might) have an effect on the things it's effecting”
- Ex: Pullback



Subcategories in Ontological Logs

From an abstract perspective, the labels "Chair" and "Support" and "Me" doesn't really mean anything unless I've defined them

Subcategories in Ontological Logs



If we expand the ontological log we might be able to capture some more structure

Some Simple Human Experimentation

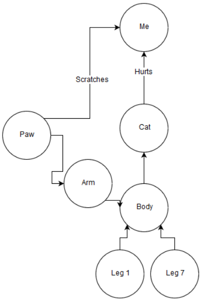
Cat

Forest

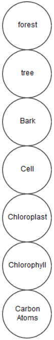
Topology

Some Simple Human Experimentation

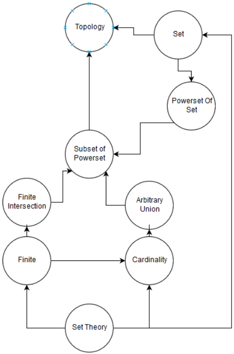
Cat



Forest



Topology



This should be minor evidence that the process of "expanding your ontological log" is maybe something that humans do to represent knowledge

Ontological Expansions : Small Subcategories

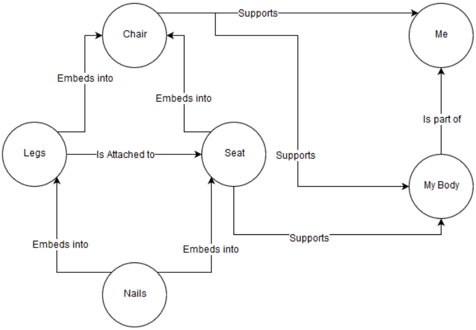
Def : Sm($\mathfrak{C}$)

Let $\mathfrak{C}$ be a category

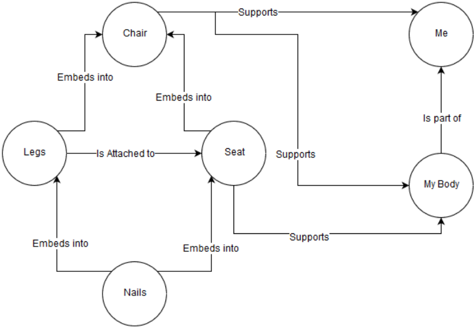
- A **small subcategory** of $\mathfrak{C}$ is a functor $S : I \to \mathfrak{C}$, for I a small category.
- For two small subcategories, define the set $Hom_{\mathfrak{C}}(S, S') = \{f : S(i) \to S'(j) | i \in I, j \in J\}$
- A **submorphism** $\mathcal{F} : S \to S'$ is a subset $\mathcal{F} \subseteq Hom_{\mathfrak{C}}(S, S')$

Example


Chair
Supports
Me


Chair
Supports
Me
Embeds into
Embeds into
Is part of
Legs
Is Attached to
Seat
Supports
My Body
Embeds into
Embeds into
Supports
Nails

Question: How to get from picture one to picture two?



Answer(?) : Ontological Generators

Naively:

Def : Ontological Generator

An **Ontological Generator** is a functor $OG : \mathfrak{C} \to Sm(\mathfrak{C})$

- For an object $X \in \mathfrak{C}$ call $OG(X)$ the "Ontological Expansion of X"
- This is great, but we want to be able to use it.
- A stronger definition is needed.

Current Def : OG

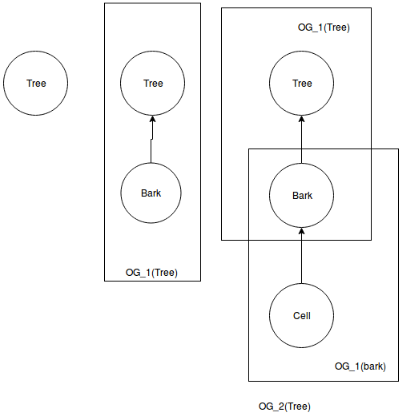
Def : OG

Let $(\$, \otimes)$ be a small monoidial category, An **Ontological Generator** is a parameterized functor $OG : \$ \to (\mathfrak{C} \downarrow sm(\mathfrak{C}))$ such that:

- **Ontological Composition** $OG_s \,\text{ꝯ}\, OG_{s'} = OG_{s \otimes s'} : \mathfrak{C} \to sm(\mathfrak{C})$
- **Colimit is a section** $Colim \circ OG_s = Id_{\mathfrak{C}}$
- **Local Measurement** For all $s \in \$, X \in \mathfrak{C}$ $OG_s(X)$ is a site

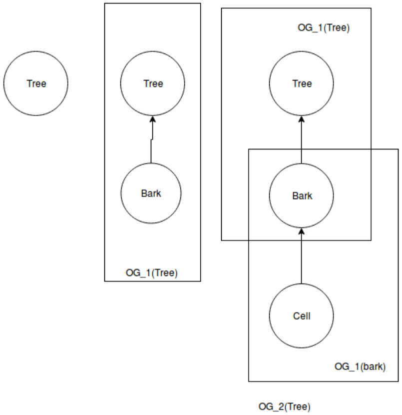
Note: You can just think of an OG as a collection of naive ontological expansions $OG_s : \mathfrak{C} \to sm(\mathfrak{C})$

OG composition (Example Picture)



We want to be able to compose expansions in a controlled and meaningful way

OG composition (Example Picture)



We want to be able to compose expansions in a controlled and meaningful way

Anisotropy: Seeing the forest for the trees (as opposed to the carbon atoms)

OG composition

Realize $Sm : \mathfrak{Cat} \to \mathfrak{Cat}$ as a functor:

- $Sm(\mathfrak{C})$ has already been defined.
- For $F : \mathfrak{C} \to \mathfrak{D}$, we want $Sm(F) : Sm(C) \to Sm(D)$.

This is simple, for $S : I \to \mathfrak{C} \in Sm(C)$ $Sm(F)(S) = F \circ S$
for $\mathcal{F} : S \to S'$, $Sm(F)(\mathcal{F}) = \{F(f) | f \in \mathcal{F}\}$

OG Composition

Let $OG_1, OG_2 : \mathfrak{C} \to sm(\mathfrak{C})$, define
$OG_1 \fatsemi OG_2 = colim \circ Sm(OG_1) \circ OG_2$
i.e. :

$$\mathfrak{C} \overset{OG_2}{\to} sm(\mathfrak{C}) \overset{sm(OG_1)}{\to} sm(sm(\mathfrak{C})) \overset{colim}{\to} sm(\mathfrak{C})$$

Current Def : OG

Def : OG

Let $(\$, \otimes)$ be a small monoidial category, An **Ontological Generator** is a parameterized functor $OG : \$ \to (\mathfrak{C} \downarrow sm(\mathfrak{C}))$ such that:

- **Ontological Composition** $OG_s \text{୨} OG_{s'} = OG_{s \otimes s'} : \mathfrak{C} \to sm(\mathfrak{C})$
- **Colimit is a section** $Colim \circ OG_s = Id_{\mathfrak{C}}$
- **Local Measurement** For all $s \in \$, X \in \mathfrak{C}$ $OG_s(X)$ is a site

Colimit is a Section

Let's first interpret what $Colim \circ OG_s$ even means.

Colimit is a Section

Let's first interpret what $Colim \circ OG_s$ even means.

- $OG_s(X) : I \to \mathfrak{C}$

Colimit is a Section

Let's first interpret what $Colim \circ OG_s$ even means.

- $OG_s(X) : I \to \mathfrak{C}$
- We can take the colimit of this functor

Colimit is a Section

Let's first interpret what $Colim \circ OG_s$ even means.

- $OG_s(X) : I \to \mathfrak{C}$
- We can take the colimit of this functor
- The statement $Colim \circ OG_s = Id_X$ is the statement that $Colim(OG_s(X)) = X$

Colimit is a Section

Let's first interpret what $Colim \circ OG_s$ even means.

- $OG_s(X) : I \to \mathfrak{C}$
- We can take the colimit of this functor
- The statement $Colim \circ OG_s = Id_X$ is the statement that $Colim(OG_s(X)) = X$

That is, we can recover the original object X from its ontological expansion

Colimit is a Section

Let's first interpret what $Colim \circ OG_s$ even means.

- $OG_s(X) : I \to \mathfrak{C}$
- We can take the colimit of this functor
- The statement $Colim \circ OG_s = Id_X$ is the statement that $Colim(OG_s(X)) = X$

That is, we can recover the original object X from its ontological expansion
"An object isn't the sum of its parts, but the colimit of its ontological expansions"

Example: The original OG, O(X)

Note that if we let $\$ = *$ and $OG_* = O : \mathfrak{T}op \to sm(\mathfrak{T}op)$. Then O(X) becomes an ontological generator.

Example: The original OG, O(X)

Note that if we let $\$ = *$ and $OG_{*} = O : \mathfrak{Top} \to sm(\mathfrak{Top})$. Then O(X) becomes an ontological generator.

- for $f : X \to Y$ let $O(f) = \mathcal{F} = \{f|_U \to V | V \supseteq f(U)\}$

Example: The original OG, O(X)

Note that if we let $\$ = *$ and $OG_* = O : \mathfrak{T}op \to sm(\mathfrak{T}op)$. Then O(X) becomes an ontological generator.

- for $f : X \to Y$ let $O(f) = \mathcal{F} = \{f|_U \to V | V \supseteq f(U)\}$
- **Ontological Composition** is trivial
- **Colim is a section** $Colim(O(X)) = \bigcup O(X) = X$
- **Local Measurement** $O(X)$ is a **site**, as seen before.

Future Work : Local Measurement

- We want to be able to deduce properties about an object

Future Work : Local Measurement

- We want to be able to deduce properties about an object
- If we expand enough, we might be able to get to some "atomic subobjects"

Future Work : Local Measurement

- We want to be able to deduce properties about an object
- If we expand enough, we might be able to get to some "atomic subobjects"
- We want to be able to build properties hierarchically from these "atoms"

Future Work : Local Measurement

- We want to be able to deduce properties about an object
- If we expand enough, we might be able to get to some "atomic subobjects"
- We want to be able to build properties hierarchically from these "atoms"
- We should be able to do this using the sheaf condition

Future Work : Local Measurement

- We want to be able to deduce properties about an object
- If we expand enough, we might be able to get to some "atomic subobjects"
- We want to be able to build properties hierarchically from these "atoms"
- We should be able to do this using the sheaf condition

In a terrible way, we can deduce your actions from the physics of the entirety of your atoms

Future Work : Local Measurement

- We want to be able to deduce properties about an object
- If we expand enough, we might be able to get to some "atomic subobjects"
- We want to be able to build properties hierarchically from these "atoms"
- We should be able to do this using the sheaf condition

In a terrible way, we can deduce your actions from the physics of the entirety of your atoms
In a less terrible way, we can deduce the physics of your cell parts from your atoms, your cells from your cell parts, your organs from your cells and then you from your organs.

OG-Sheaves / Measurement

That is, we want a definition along the lines of:

Current Work : OG-Sheaves / Measurement

A **Measurement** of an ontological generator OG, is a collection of sheaves $P_{s,X} : OG_s(X) \to \mathfrak{D}$

OG-Sheaves / Measurement

That is, we want a definition along the lines of:

Current Work : OG-Sheaves / Measurement

A **Measurement** of an ontological generator OG, is a collection of sheaves $P_{s,X} : OG_s(X) \to \mathfrak{D}$

The real question is then: What should be used for $\mathfrak{D}$?

Future Work : Intuition, actions and measurements

Furthermore, we want to change our intuition upon viewing an ontology.
First some interpretations:
$\mathfrak{C} \implies$ objects
$sm(\mathfrak{C}) \implies$ ontological representations
$sm(sm(\mathfrak{C})) \implies$ categories of ontological representations.
So the idea is that an object $\mathscr{S} \in sm(sm(\mathfrak{C}))$ should be an **intuition** about the universe $\mathfrak{C}$. An ontological updator tells us how to change our intuitions

Future Work : Intuition, actions and measurements

Furthermore, we want to change our intuition upon viewing an ontology.
First some interpretations:
$\mathfrak{C} \implies$ objects
$sm(\mathfrak{C}) \implies$ ontological representations
$sm(sm(\mathfrak{C})) \implies$ categories of ontological representations.
So the idea is that an object $\mathscr{S} \in sm(sm(\mathfrak{C}))$ should be an **intuition** about the universe $\mathfrak{C}$. An ontological updator tells us how to change our intuitions

Def : Ontological Updator

$U : sm(\mathfrak{C}) \to endFunc(sm(sm(\mathfrak{C})))$

Deep Conversations as Updators

- Describing an object X is akin to expanding it $OG_s(X)$

Deep Conversations as Updators

- Describing an object X is akin to expanding it $OG_s(X)$
- Combined with an ontological updator, then $U(OG_s(X))$ tells you how to change your intuition. I.e. $U(OG_s(X))(\mathscr{S})$ is your new intuition after listening to $OG_s(X)$

Deep Conversations as Updators

- Describing an object X is akin to expanding it $OG_s(X)$
- Combined with an ontological updator, then $U(OG_s(X))$ tells you how to change your intuition. I.e. $U(OG_s(X))(\mathscr{S})$ is your new intuition after listening to $OG_s(X)$

"How someone's ontological expansion changed your intuition"

Deep Conversations as Updators

- Describing an object X is akin to expanding it $OG_s(X)$
- Combined with an ontological updator, then $U(OG_s(X))$ tells you how to change your intuition. I.e. $U(OG_s(X))(\mathscr{S})$ is your new intuition after listening to $OG_s(X)$

"How someone's ontological expansion changed your intuition"

Example: You ask someone to tell you some macroscopic properties of a cat, they tell you that a cat has a tail. You say to yourself, "Wow, I already knew that" and so you add to your ontology that the person you are talking to must think you're pretty dull.

Issues

1) In defining the spiral product, we are implicitly making the assumption that $sm(\mathfrak{C})$ is cocomplete. But it seems like this assumption is satisfied by the cocompleteness of $\mathfrak{C}$.

2) $Colim(OG_s(X))$ works fine if just considering the subcategory $OG_s(X)$, however we'd like to have $Colim \circ OG_s$ a functor. To this end we search for a "correct" embedding functor $sm(\mathfrak{C}) \to (sm\mathfrak{C}at \downarrow \mathfrak{C})$ completing the chain:

$$\mathfrak{C} \overset{OG_s}{\to} sm(\mathfrak{C}) \overset{i}{\to} (sm\mathfrak{C}at \downarrow \mathfrak{C}) \overset{Colim}{\to} \mathfrak{C}$$