# CATEGORY THEORY

# GENERAL ABSTRACT NONSENSE

MARTIN HEUSCHOBER

VIENNA, APR. $13^{TH}$ 2016,

# FOUNDATION

# MATH

Category theory is a bit complicated for two reasons (at least):

1. The topic evolved in a quite advanced field in mathematics, therefore it usually is mentioned late (I had to study 5-7 Semesters to find a seminar about that topic).
2. The foundation needed to speak about it properly one needs class theory instead of set theory.

# AD 1

The topic where it evolved was 'Algebraic Topology', in the search of invariants of topological spaces, one discovered that you can associate a group with said space. Which is what we call now a Functor.

# AD 2

The minimal knowledge about class theory one needs is that we distinguish between two classes of containers - sets and classes, where classes are collections of magnitude beyond everything. Think of the `Set` of all `Sets` or the Russel's Paradox, the `set of all sets that contain itself`.

In the following slides we will for example talk about the class of all vector spaces, which is a class. If you accept that the set of all sets is a class, then note that every vector space is determined by the set of its base vectors. Thus you get the class of all vector spaces is not a set.

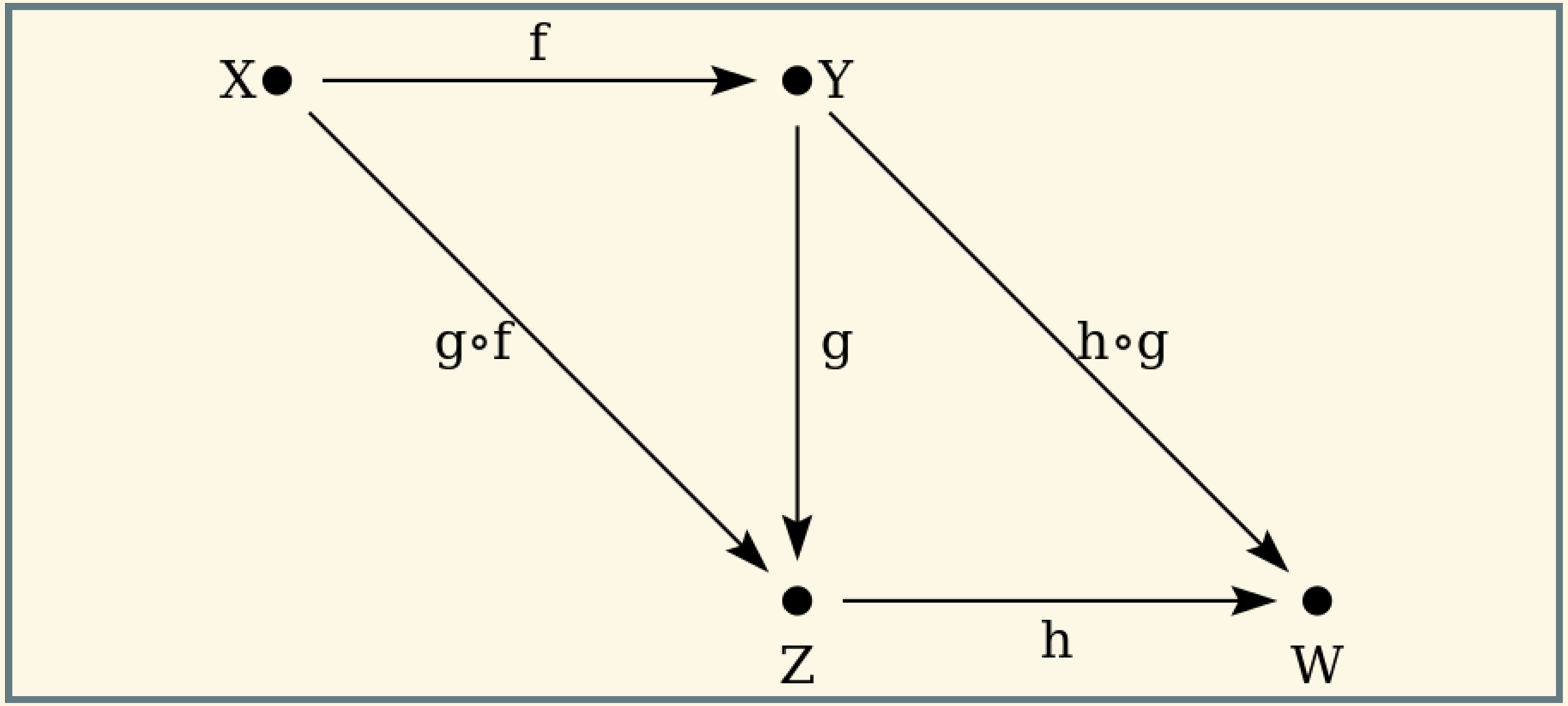
During all slides I will refer to classes that can be modeled by ordinary set theory as `sets` or `small classes`, and collections that cannot as `classes` or `proper classes`.

# CATEGORY

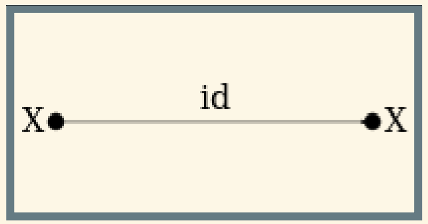
# DEFINITION

A category $\mathcal{C}$ is

1. A class with members called **Objects** $obj(\mathcal{C})$
2. For every object $X$ there exists a unique morphism $id_X : X \to X$ *Note: If the object is unambiguous we often omit the subscript X.*
3. For every two objects $X, Y$ we have a set $\mathcal{C}(X, Y)$ with members called **morphisms from** $X$ **to** $Y$ such that we call two morphisms equal, if they have the same input set (=domain), output set (=codomain) and for each input the same output is generated.
4. For all objects $X, Y$ and $Z$ we have a map
$$\circ : \mathcal{C}(Y, Z) \times \mathcal{C}(X, Y) \to \mathcal{C}(X, Z)$$
called **composition** such that the law of associativity
$$f \circ (g \circ h) = (f \circ g) \circ h$$
holds.




any path from $X$ to $W$ must be equal in a category.



# EXAMPLES SMALL

- every monoid $\mathcal{M}$ with composition being the monoidal $\bullet$ and identity given by the identity element of the monoid
- every set $\mathcal{S}$ with composition being good old function composition $\circ$ and identity given by the identity function.

# EXAMPLES MEDIUM

Every set of sets $\mathcal{P}$ with arrows being given by set inclusion $\subseteq$.

So for $A \subseteq B$ we have $A \to B$

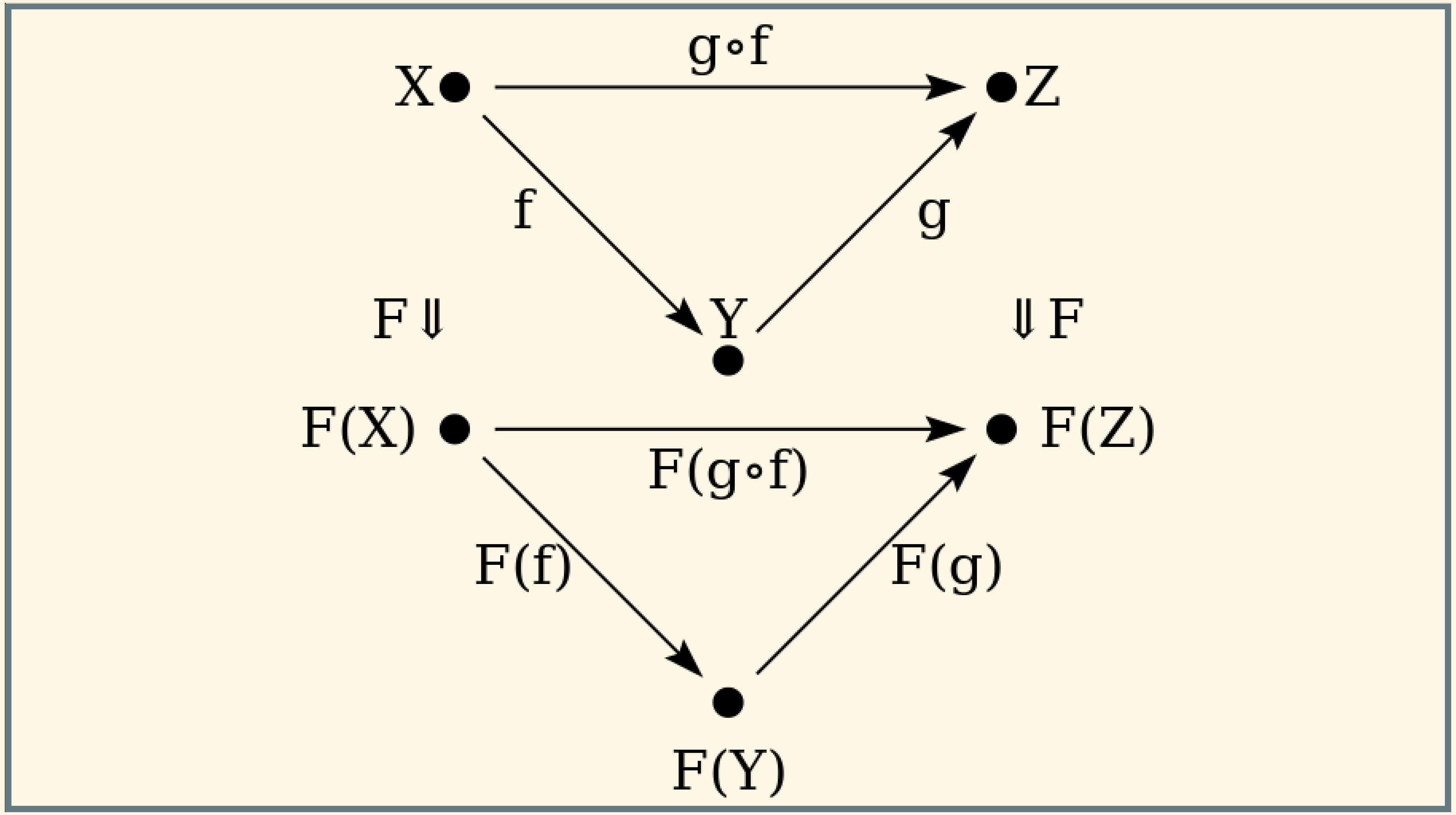
So $A \subseteq B \subseteq C$ we get $A \to B \to C$ of course by transitivity of "$\subseteq$" we get $A \subseteq C$.

# EXAMPLES BIG

- $\mathcal{S}et$ … the category of all mathematical sets with functions between them
- $\mathbb{R} - \mathcal{V}ector\mathcal{S}pace$ … the category of all linear spaces over the field of real numbers, with arrows being linear functions
- $\mathcal{PO} - \mathcal{S}ets$ … the category of partially ordered sets with arrows being given by the inclusion

# CONNECTING CATEGORIES

# FUNCTOR




Functor

# EXAMPLES

The fundamental group of a topological space

$$\pi_1 : \mathcal{T}op \to \mathcal{G}rp$$

# EXAMPLES SMALL

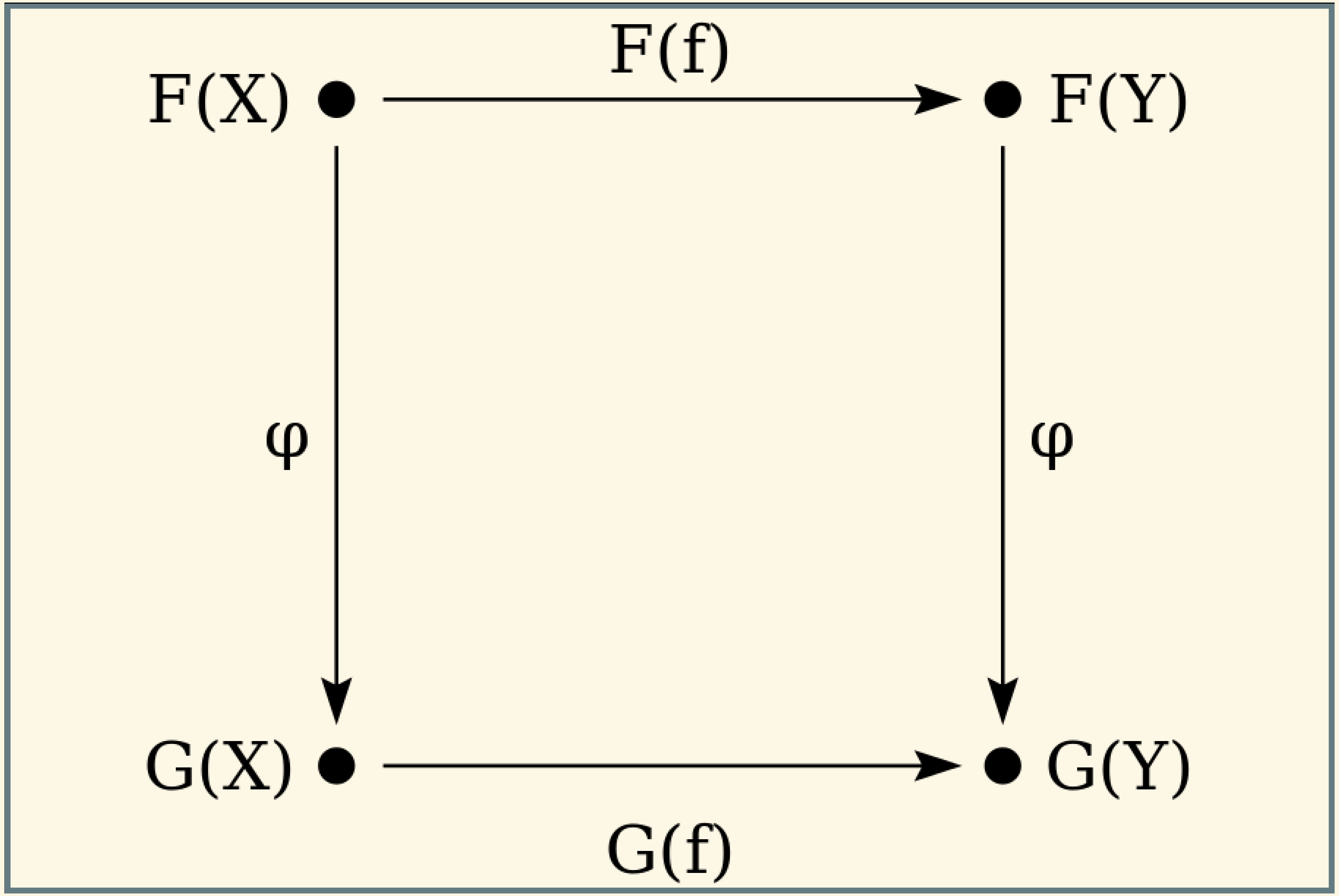
- every homomorphism between two monoids $\mathcal{M}, \mathcal{N}$ can be viewed as a functor
- thus `length :: [a] -> Int` is a functor
- every type `a` we get `[a]` as a functor
- I think this is the same as the free monoid over a set $\mathcal{S}$

# EXAMPLES BIG

- for every (small) category we have the forgetful functor
$$F : \mathcal{C} \to \mathcal{S}et$$
- for every algebraic structure we have a functor from a more specialised into a general structure - for example every group is a monoid, therefore we have a functor $\mathcal{G}rp \to \mathcal{M}on$

# NATURAL TRANSFORMATION

Of course one can make the existing theory a bit more interesting and associate functors with each other - we call a map between two functors $F, G$ a **natural transformation**, if for all objects $X$ of $\mathcal{C}$ we get a morphism $\varphi_X$, such that for all morphisms $f : X \to Y$ the following diagram commutes.


F(f)
F(X)
F(Y)
φ
φ
G(X)
G(Y)
G(f)

# EXAMPLES - PLEASE

- `flatten :: Tree a  -> [a]`
- ??

# CONCEPTS

# DUALITY

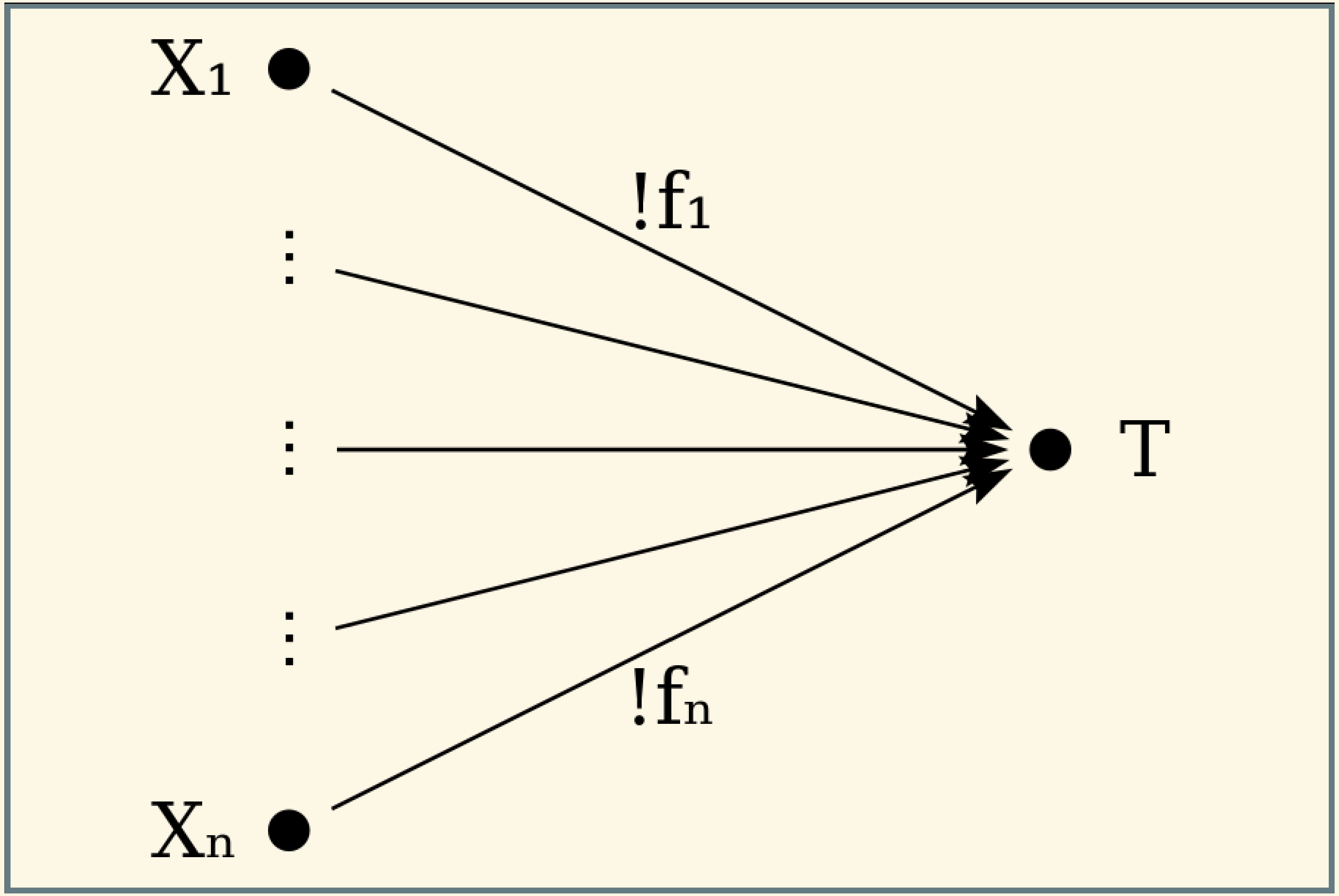
For every category $C$ we have the opposite category $\mathcal{C}^{op}$, where the composition is defined as $f \circ^{op} g = g \circ f$, we get it by simply reversing all arrows. For each 'concept' we thus get a 'concept' in the opposite category - we call such concepts **dual** and prefix the existing concept with 'co', as for example in *co*functor.

$$
\begin{tikzcd}
X & Y \arrow[l, "f"'] \\
& Z \arrow[u, "g"'] \arrow[ul, "f\circ g"]
\end{tikzcd}
$$

# SPECIAL OBJECTS

# TERMINAL OBJECTS

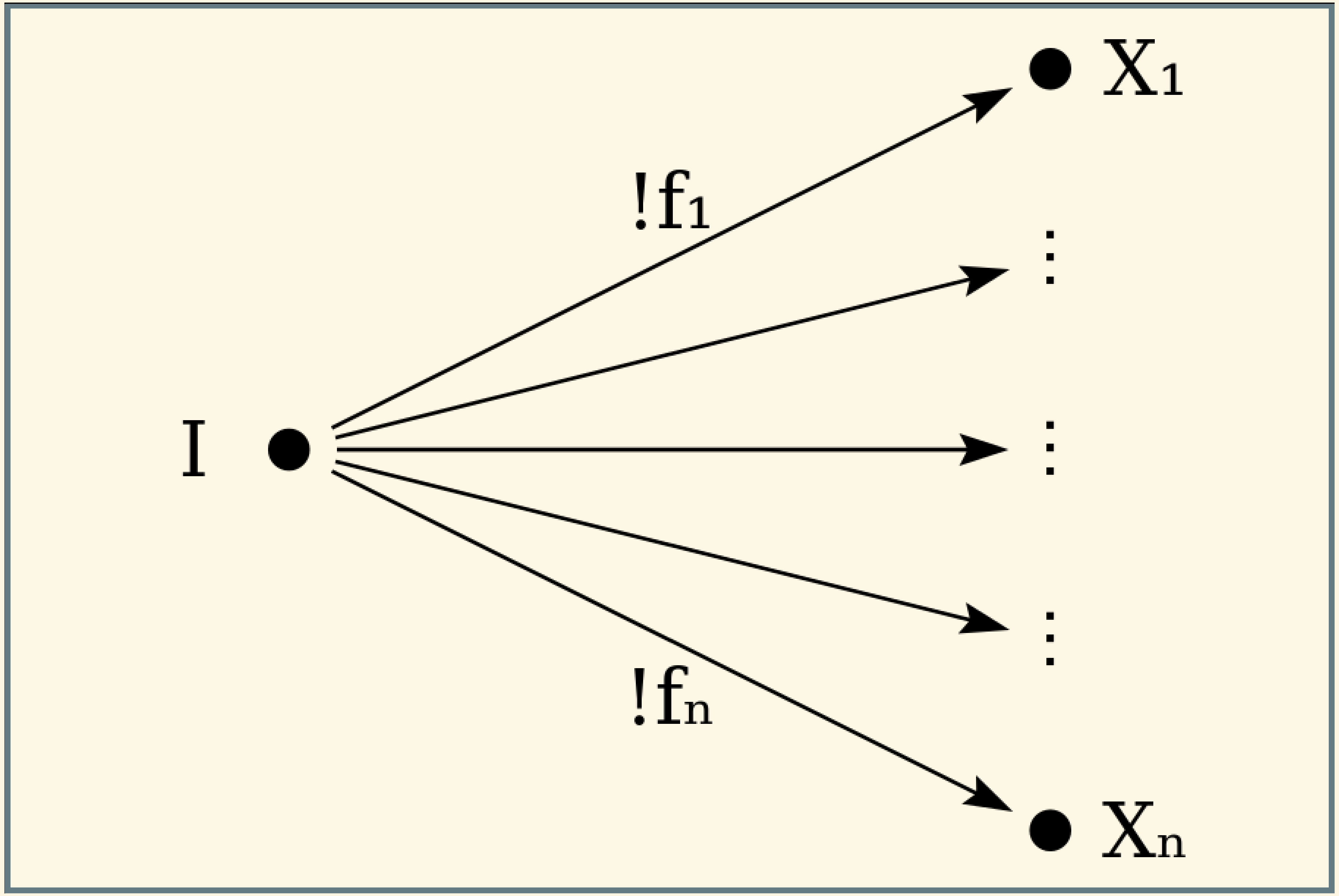
An object $T$ in a category $\mathcal{C}$ is called **terminal**, if for every object $X$ in this category we have a unique morphism $f_X : X \to T$.




Note: The index $n$ should not indicate that there are finitely many objects but just that there are many.

# INITIAL OBJECTS

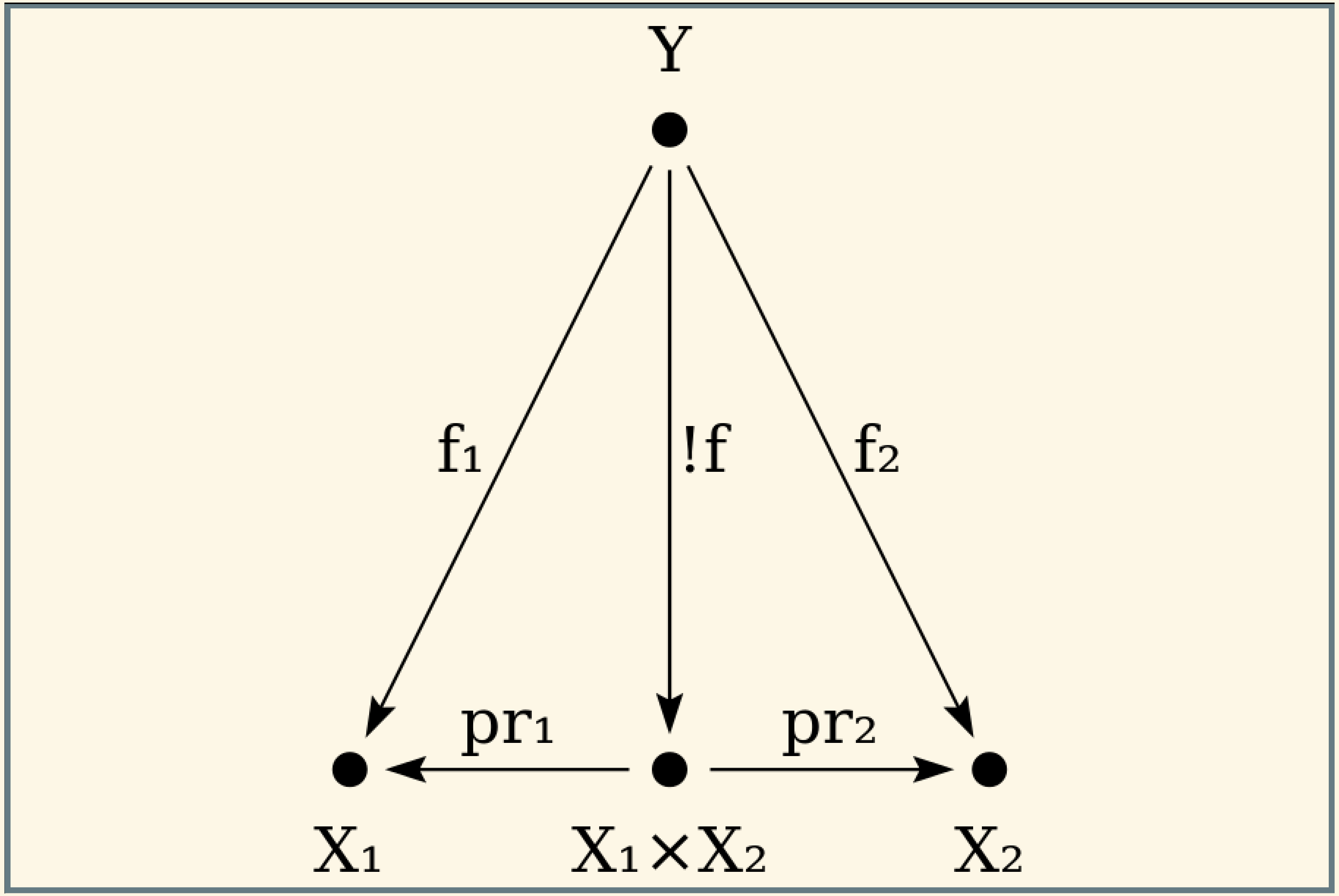
An object $I$ in a category $\mathcal{C}$ is called **initial**, if for every object $X$ in this category we have a unique morphism $f_X : I \to X$.




Note: The index $n$ should not indicate that there are finitely many objects but just that there are many.

# PRODUCT OBJECTS

An object in a category is called **product** of $X_1$ and $X_2$, if it has two morphisms $pr_1$ and $pr_2$, and for all other objects $Y$ and morphisms $f_1 : Y \to X_1$ and $f_2 : Y \to X_2$ we get a unique map $f$ from $Y$ to this object. We write this object $X_1 \times X_2$.


$Y$
$f_1$
$!f$
$f_2$
$pr_1$
$pr_2$
$X_1$
$X_1 \times X_2$
$X_2$

# HASKELL

- $pr_1$ = `fst`
- $pr_2$ = `snd`

```
import Control.Arrow
```

- `(***) :: Arrow a => a b c -> a b' c' -> a (b,b') (c,c')`
    - $f$ = `f₁ *** f₂`

# AND WITH DUALITY

# SUM OBJECTS

An object in a category is called **coproduct** or **sum** of $X_1$ and $X_2$, if it has two morphisms $\iota_1$ and $\iota_2$, and for all other objects $Y$ and morphisms $f_1 : X_1 \to Y$ and $f_2 : X_2 \to Y$ we get a unique map $f$ from this object to $Y$. We write this object $X_1 \oplus X_2$.

$$
\begin{tikzcd}
 & Y & \\
X_1 \arrow[ur, "f_1"] \arrow[r, "\iota_1"'] & X_1 \oplus X_2 \arrow[u, "!f"'] & X_2 \arrow[ul, "f_2"'] \arrow[l, "\iota_2"]
\end{tikzcd}
$$

# HASKELL

- $\iota_1$ = `Left`
- $\iota_2$ = `Right`

```
import Data.Either
```

- `either :: (a -> c) -> (b -> c) -> (Either a b) -> c`
- $f$ = `either f₁ f₂`

Note that every sum/product is unique up to isomorphism.

# PROOF

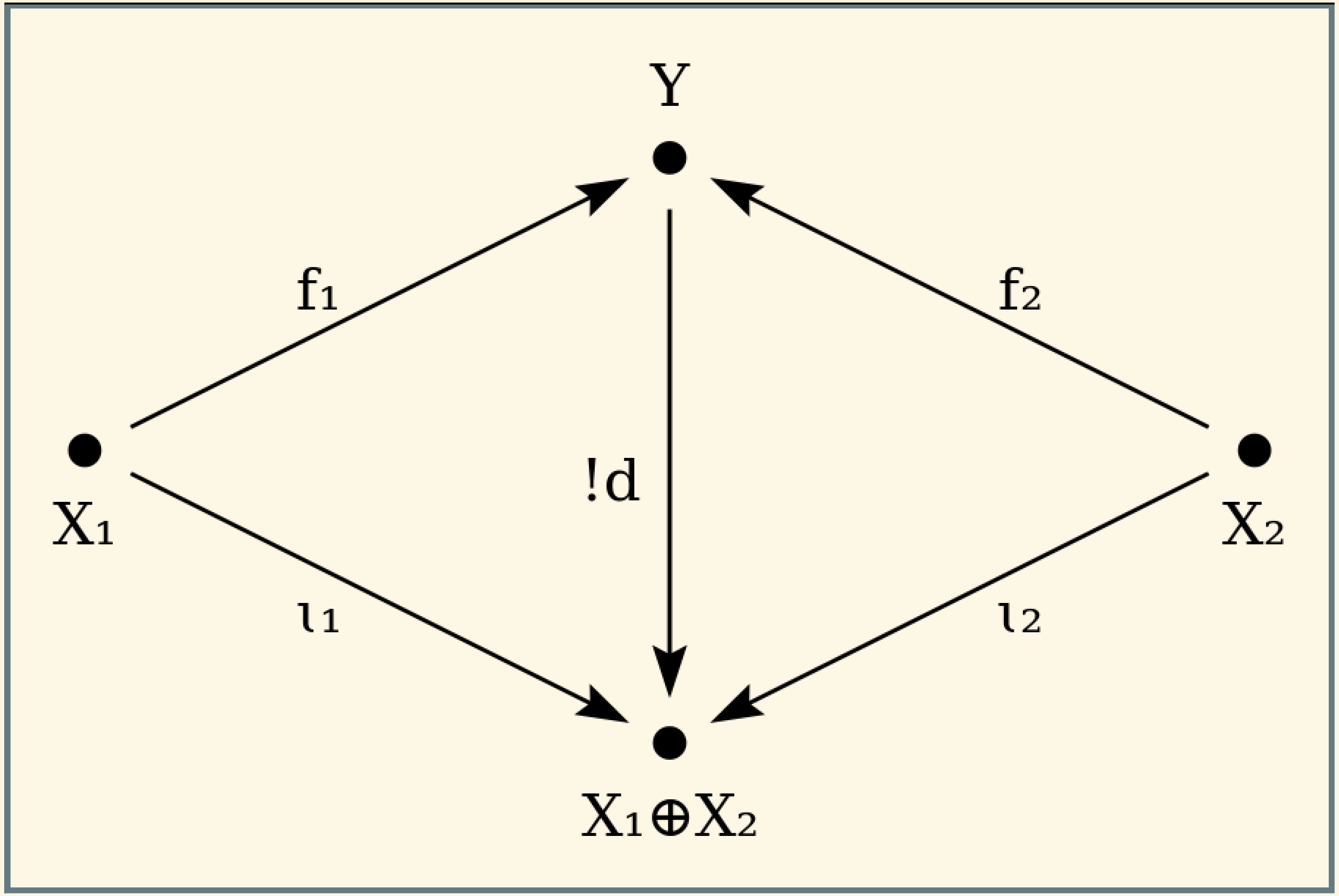
$$
\begin{tikzcd}
X_1 \arrow[r, "l_1"] & X_1 \boxplus X_2 & X_2 \arrow[l, "l_2"'] \\
X_1 \arrow[r, "l_1"'] & X_1 \oplus X_2 & X_2 \arrow[l, "l_2"]
\end{tikzcd}
$$

Suppose we had two objects $X_1 \boxplus X_2$ and $X_1 \oplus X_2$ - with the universal property of the sum.

# PROOF

$$
\begin{tikzcd}
 & X_1 \boxplus X_2 & \\
X_1 \arrow[ur, "\iota_1"] \arrow[dr, "f_1"'] & & X_2 \arrow[ul, "\iota_2"'] \arrow[dl, "f_2"] \\
 & Y \arrow[uu, "!u"'] &
\end{tikzcd}
$$

# PROOF

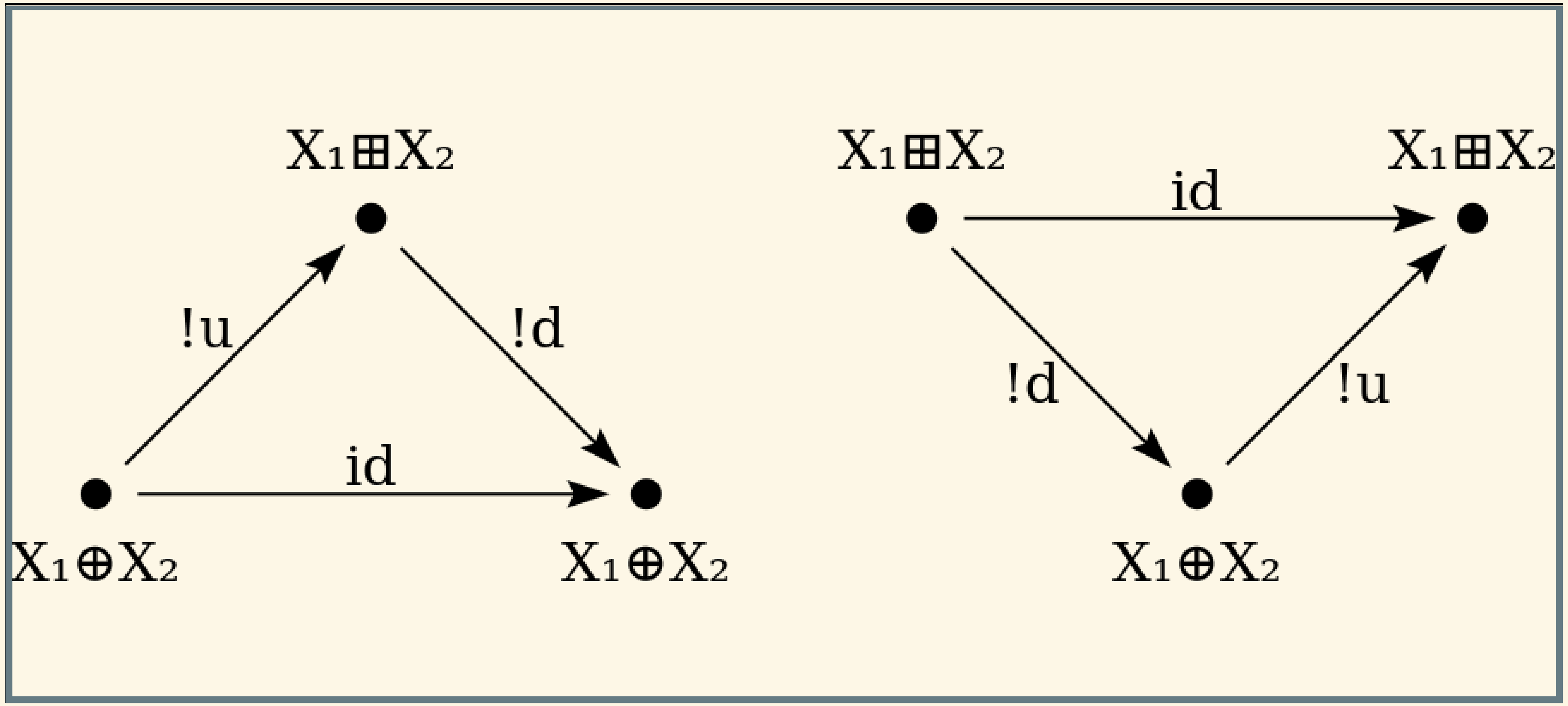


# PROOF

$$
\begin{tikzcd}
 & X_1 \boxplus X_2 \arrow[dd, leftrightarrow, "\mathrm{id}"] & \\
X_1 \arrow[ur, "\iota_1"] \arrow[dr, "\iota_1"'] & & X_2 \arrow[ul, "\iota_2"'] \arrow[dl, "\iota_2"] \\
 & X_1 \boxplus X_2 &
\end{tikzcd}
$$

$$
\begin{tikzcd}
& X_1 \oplus X_2 \arrow[dd, leftrightarrow, "\mathrm{id}"] & \\
X_1 \arrow[ur, "\iota_1"] \arrow[dr, "\iota_1"'] & & X_2 \arrow[ul, "\iota_2"'] \arrow[dl, "\iota_2"] \\
& X_1 \oplus X_2 &
\end{tikzcd}
$$

# PROOF




$u \circ d = id$

# AWESOME THINGS I KNOW NOTHING OF

# YONEDA LEMMA

# KAN-EXTENSIONS

HASK

# IS NOT A CATEGORY

# WHY?

BECAUSE OF

# UNDEFINED

see haskell-wiki