Computational methods in category theory

Simon Forest

LIS, Aix-Marseille Université

February 13, 2025

Notions of computations

To "compute" can have **different meanings** depending on context

General definition: to construct **something** using a sequence of **known operations**.

- "something": numbers, mathematical objects, instances of datastructures, *etc.*
- "known operations": addition, colimits, `while` loops, *etc.*

In the CS meaning, different **levels** of computability or **computational methods**

- **algorithm** (gold standard): fully terminating procedure on all instances
- **procedure**: might terminate or not
- **proof assistant**: ask the user for the next step of computation, check that the steps are sound

Computable functions

There is different models for the notion of computable functions, like the one of recursive functions.

Recursive functions: subclasses Rec_k for $k \geq 0$ of partial functions $\mathbb{N}^k \to \mathbb{N}$ s.t.

- constant functions $(n_1, \dots, n_k) \mapsto c$ are in Rec_k
- projections $(n_1, \dots, n_k) \mapsto n_i$ are in Rec_k
- closed by composition and recursion

But computation is not just about natural numbers. What about other structures?

We can still replace $\mathbb{N}$ by **datastructures** (`lists`, `maps`, *etc.*) but this is just syntax for $\mathbb{N}$.

Can we talk about other mathematical objects with $\mathbb{N}$?

Computing on mathematical objects

Given a "universe" (a set, a category, *etc.*) of objects $\mathcal{U}$, an **encoding** is a subset $E \subseteq \mathbb{N}$ together with a mapping

$$\begin{array}{ccc} E & \to & \mathcal{U} \\ e & \mapsto & [\![e]\!] \end{array}$$

An object $X \in \mathcal{U}$ is **encoded** by E when there is $e \in E$ and an "equivalence" $X \simeq [\![e]\!]$.

Examples: the finite sets are encoded by $\mathbb{N}$ through the mapping

$$n \quad \mapsto \quad \{0, \dots, n-1\}$$

Computing on mathematical objects

Once such an encoding is given, one can ask what operations can be computed with it, as recursive functions.

Examples with sets:

Disjoint union	Product
$(S,T) \mapsto S \coprod T$	$(S,T) \mapsto S \times T$
$(n,m) \mapsto n+m$	$(n,m) \mapsto nm$

Using this kind of encoding, one can wonder what is computable among mathematical structures.

Category

Category theory: categories, functors, natural transformations, their constructions and properties.

Given a category $\mathcal{C}$, examples of things that we want to know:

- is $\mathcal{C}$ complete or cocomplete?
- is $\mathcal{C}$ closed?

Given a functor $F\colon \mathcal{C} \to \mathcal{D}$, examples of things that we want to know:

- does F preserve limits or colimits?
- is F part of an adjunction?

Can we find encodings enabling computational methods for these problems?

Outline

Outline

Computing with f.p. categories

Computing with **Set**

Computing with presheaves

Encoding standard categories

Encoding functors

Method for left adjointness

A method for cartesian closure

Applications

Bonus slides

Computing with presentations

When computing with mathematical objects, the standard way to go is with **presentations**.

Computing with presentations

Given a group G, one can express it using a **presentation** $\langle S \mid E \rangle$

- $\mathbb{Z}^2 \cong \langle \{a, b\} \mid ab = ba \rangle$
- $\mathbb{Z}/n\mathbb{Z} \cong \langle \{a\} \mid a^n = 1 \rangle$
- ...

When such a presentation is finite, one can easily describe it to a computer

```
let Z2   = group(gens = {a,b}, eqs = {([a;b],[b;a])})
let Z/3Z = group(gens = {a},   eqs = {([a;a;a],[])})
```

Computing with presentations

Using this encoding or others, several algorithms were introduced, notably:

- **Todd–Coxeter algorithm**: coset enumeration
- **Schreier–Sims algorithm**: find the order of a permutation group

We can also present **morphisms** between presented groups and make computations on them.

Presentations for categories

Remember that categories, technically, are **algebraic structures** (*essentially*).

2 sorts:

$$\mathbf{C_0} \quad \text{and} \quad \mathbf{C_1}$$

4 operations:

$$\mathrm{id}\colon \mathbf{C}_0 \to \mathbf{C}_1 \qquad \partial^-\colon \mathbf{C}_1 \to \mathbf{C}_0 \qquad \partial^+\colon \mathbf{C}_1 \to \mathbf{C}_0 \qquad c\colon \mathbf{C}_1 \times_0 \mathbf{C}_1 \to \mathbf{C}_1$$

They thus admit a notion of presentation.

Computational representations

Example: one can consider a category C with

- objects u, v, w
- generating arrows $a\colon u \to v$, $b\colon v \to w$ and $c\colon u \to v$

$$
\begin{tikzcd}
u \arrow[dd, "a"'] \arrow[dr, "c"] & \\
& w \\
v \arrow[ur, "b"'] &
\end{tikzcd}
$$

Computational representations

Also a category D with

- objects x, y, z
- generating arrows $d\colon x \to y$ and $e\colon y \to z$

$$\begin{tikzcd} u \arrow[dd, "a"'] \arrow[dr, "c"] & \\ & w \\ v \arrow[ur, "b"'] & \end{tikzcd} \qquad \begin{tikzcd} x \arrow[r, "d"] & y \arrow[r, "e"] & z \end{tikzcd}$$

Computational representations

Then one can consider the functor F such that

$$F(u) = x \qquad F(v) = y \qquad F(w) = z$$
$$F(a) = d \qquad F(b) = e \qquad F(c) = d * e$$

$$\begin{tikzcd} u \arrow[d, "a"'] \arrow[dr, "c"] & \\ v \arrow[r, phantom] & w \\ \end{tikzcd} \xrightarrow{F} \begin{tikzcd} x \arrow[r, "d"] & y \arrow[r, "e"] & z \end{tikzcd}$$

Computational representations

Such data can be given to a computer.

```
A := category {
  obj := {u,v,w},
  arr := {a : u => v, b : v => w, c : u => w}
}
B := category {
  obj := {x,y,z},
  arr := {d : x => y, e : y => z}
}
F := functor A => B {
  u -> x, v -> y, w -> z,
  a -> d, b -> e, c -> d * e
}
```

Computational representations

Such encoding allows considering the computability of several construction or property on **finitely presented categories**.

But the categories described this way feels **very artificial**.

What about real categories like $\mathbf{Set}$, $\mathbf{Grp}$, *etc.* and associated functors.

Still, some successes were obtained with this encoding:

- **solution for the word problem** on morphisms based on rewriting
- **computation of Left Kan extensions** (some at least) [Carmody,Walters]
    - generalisation of Todd–Coxeter algorithm for groups

Outline

The category **Set**

The category **Set** of sets and functions is a standard example of categories.

The situation of **Set** is already different from an f.p. category

- its classes of objects and morphisms is not finite (not even sets!)
- it is *morally* only defined up to equivalence of category

Let's see what we can compute *inside* **Set**.

Encoding finite sets

Finite sets can easily be encoded as follows.

Type of integers: `type nat = Zero | Succ of nat`

Type of elements: `type el = El of nat`

Type of sets: `type set = el list`

- **but**: need to filter lists with duplicates
- more efficient: use `Set` Module to create Sets of `el`

Mapping:

$$[\texttt{El }1; \texttt{El }5; \texttt{El }7; \texttt{El }42] \qquad \mapsto \qquad \{1, 5, 7, 42\}$$

Encoding functions between sets

Functions between finite sets can be encoded as follows.

Type of functions (1st try): `type sfun = el -> el`

- difficult to define and inspect
- more generally, not efficient

Type of functions (2nd try): `type sfun = (el * el) list`

- easier to define and inspect
- still a bit inefficient
- better: use `Map` Module to create `Map`s of (`el`,`el`)-bindings

Assuming sets S, S' encoded by `[El 1;El 2]` and `[El 4;El 5; El 6]`, we have a mapping:

$$[(\texttt{El }1,\ \texttt{El }6);(\texttt{El }2,\texttt{El }4)] \qquad \mapsto \qquad f \quad = \quad S \cong \{1,2\} \xrightarrow{f'} \{4,5,6\} \cong S'$$

with f' defined by $f'(1) = 6$ and $f'(2) = 4$

Observations

Simple computability observations:

- From an encoding of a finite set S, the identity function id_S is encodable
- From encodings of functions $S \xrightarrow{f} S' \xrightarrow{f'} S''$, the composite $f' \circ f$ is computable

$$\texttt{(s,s')} \in \texttt{f}, \quad \texttt{(s',s'')} \in \texttt{f'} \qquad \rightsquigarrow \qquad \texttt{(s,s'')} \in \texttt{ff'}$$

Encoding graphs

An oriented (multi-)graph is a diagram

$$A \underset{\mathrm{t}}{\overset{\mathrm{s}}{\rightrightarrows}} N$$

in **Set**.

An oriented graph G induces a free category G^* of **paths** between nodes.

Encoding graphs

Finite graphs $A \overset{s}{\underset{t}{\rightrightarrows}} N$ can be encoded quite easily:

```
type graph = { nodes  : set ;
               arrows : set ;
                  src : sfun ;
                  tgt : sfun }
```

Encoding diagrams

A **diagram** (on a graph) in **Set** is a functor $d\colon G^* \to \mathbf{Set}$, for some graph G.

Given an encoding of $G = (N, A)$, diagrams on G^* **with finite sets as images** can be encoded:

```
type diagram = { obj_map : el -> set ;
                 arr_map : el -> sfun }
```

where

- `obj_map` encodes $\mathrm{Ob}(d)\colon N \to \mathrm{Ob}(\mathbf{Set})$
- `arr_map` encodes $G \to G^* \xrightarrow{d} \mathrm{Morph}(\mathbf{Set})$

Computing colimits

Recall that a colimit Q on a (general) diagram $d\colon C \to \mathcal{D}$ can be expressed as

$$\begin{tikzcd} \coprod_{f\colon i \to j \in C} d(i) \arrow[r, shift left, "{[\iota_i]_{f\colon i\to j}}"] \arrow[r, shift right, "{[\iota_j \circ f]_{f\colon i\to j}}"'] & \coprod_{i \in C} d(i) \arrow[r, dashed, "q"] & Q \end{tikzcd}$$

When $C = G^*$, we can replace the left coproduct by $\coprod_{f\colon i \to j \in G} d(i)$

Computing colimits

In $\mathbf{Set}$, finite coproducts

$$S \quad = \quad \coprod_{i \in I} S_i$$

of encoded sets S_i are easy to compute, together with the coprojections $S_i \to S$.

Moreover, coequalisers in $\mathbf{Set}$ can be described as

$$\begin{tikzcd} S \arrow[r, shift left, "f"] \arrow[r, shift right, "g"'] & T \arrow[r, dashed, "q"] & T_{/\sim} \end{tikzcd}$$

where $\sim$ generated by $f(s) \sim g(s)$ for every $s \in S$.

When S, T and f, g are encoded, we are able to compute $T_{/\sim}$ and q using a $\textsc{Union–Find}$ algorithm in (almost) linear time.

Computing colimits

Thus, we can compute colimits over diagrams

$$d\colon G^* \to \mathbf{Set}$$

where G is finite and d has images in finite sets.

Moreover, given a category I, together with a bijective-on-objects epimorphism

$$e\colon G^* \to I$$

where G is a finite graph, the colimit on a diagram $d\colon I \to \mathbf{Set}$ is the same as the one for $d \circ e$.

Conclusion: we can compute **finite colimits** of **finite sets** over categories I with finite sets of objects and of generating morphisms.

Computing factorisations

Given an encoded diagram $d \colon G^* \to \mathbf{Set}$, a cocone

$$(p_i \colon d(i) \to S)_{i \in G}$$

where S is finite, can be encoded.

```
type cocone = {   vertex : set ;
                  coprojs : el -> sfun }
```

- `vertex` encodes S
- `coprojs` encodes the p_i's

Computing factorisations

Given the colimit $(q_i \colon d(i) \to Q)$ where $Q = (\coprod d(i))_{/\sim}$, the **factorisation**

$$\begin{tikzcd}
\vdots & & \\
d(i) \arrow[dd, "d(f)"'] \arrow[dr, "q_i"] \arrow[drr, bend left, "p_i"] & & \\
& Q \arrow[r, dashed, "p"] & S \\
d(j) \arrow[ur, "q_j"'] \arrow[urr, bend right, "p_j"'] & & \\
\vdots & &
\end{tikzcd}$$

of the cocone $(p_i \colon d(i) \to S)_{i \in G}$ can be computed.

Indeed, each element of Q is the image of some $x \in d(i)$, so that the mappings of p can be computed as:

1) $P :=$ `empty`
2) for each i, for each $x \in d(i)$, add $(q_i(x), p_i(x))$ to P
3) return P

which gives the encoding of p as `sfun`.

Outline

Our computable framework can be easily extended to presheaves.

Encoding presheaves

Given a finitely presented category C, **finite** presheaves X on C can be encoded by

```
type presheaf = {
  obj_map : el -> set ;
  arr_map : el -> sfun
}
```

where

- `obj_map` encodes $\mathrm{Ob}(X)\colon \mathrm{Ob}(C^{\mathrm{op}}) \to \mathrm{Ob}(\mathbf{Set})$
- `arr_map` encodes the mapping $\mathrm{Morph}(X)\colon \mathrm{Morph}(C^{\mathrm{op}}) \to \mathrm{Morph}(\mathbf{Set})$ (only for the generating morphisms of C)

Alternatively, with more efficient `EMap`s, that is `Map`s of `el`:

```
type presheaf = {
  obj_map : set EMap.t ;
  arr_map : set EMap.t
}
```

Encoding morphisms

Given two encoded presheaves $X, Y \in \widehat{C}$, the morphisms $m\colon X \to Y$ (that is, the natural transformations $m\colon X \Rightarrow Y\colon C^{\mathrm{op}} \to \mathbf{Set}$) can be encoded:

```
type ps_morph = {
  psm_arr_map : el -> sfun
}
```

Alternatively, with `Maps`:

```
type ps_morph = {
  psm_arr_map : sfun EMap.t
}
```

Encoding/Computing other things

What we did for **Set** naturally extends to $\widehat{C}$.

- computing composition of morphisms
- encoding cocones
- computing colimits
- computing factorisation for cocones

Outline

Presheaf categories encompass already several examples, like $\mathbf{Set}$, $\mathbf{Grph}$, *etc.*

But we still miss some important examples: $\mathbf{Mnd}$, $\mathbf{Grp}$, *etc.*

Idea: constrain the objects of presheaf categories, in order to be more expressive

Example

Consider the theory of monoids:

1 sort:

$$\mathbf{M}$$

2 generating operations:

$$e \colon 1 \to \mathbf{M} \qquad\qquad c \colon \mathbf{M}^2 \to \mathbf{M}$$

How to monoids as presheaves?

Example

$$\mathbf{M}$$

Start with sorts as objects.

Example

$$\mathbf{1} \qquad \mathbf{M} \qquad \mathbf{M}^2$$

Add objects for the domains of the operations.

$$e\colon 1 \to \mathbf{M} \qquad c\colon \mathbf{M}^2 \to \mathbf{M}$$

Example

$$\mathbf{1} \xrightarrow{e} \mathbf{M} \xleftarrow{c} \mathbf{M^2}$$

Add the arrows for these operations.

Example

$$
\begin{tikzcd}
\mathbf{1} \arrow[r, "e"] & \mathbf{M} & \mathbf{M^2} \arrow[l, shift right=2, "\pi_L"'] \arrow[l, "c" description] \arrow[l, shift left=2, "\pi_R"]
\end{tikzcd}
$$

Add arrows for the cone projections.

Example

$$\begin{tikzcd} \mathbf{1} & \mathbf{M} \arrow[l, "e"'] \arrow[r, shift left=2, "\pi_L"] \arrow[r, "c" description] \arrow[r, shift right=2, "\pi_R"'] & \mathbf{M^2} \end{tikzcd}$$

Reverse all arrows.

Example

$$\mathbf{1} \xleftarrow{e} \mathbf{M} \;\substack{\xrightarrow{\pi_L} \\ \xrightarrow{c} \\ \xrightarrow[\pi_R]{}}\; \mathbf{M^2}$$

A monoid is then a particular **presheaf** on the above category C, *i.e.*, a functor

$$X \colon C^{\mathrm{op}} \to \mathbf{Set}$$

Example

$$\mathbf{1} \xleftarrow{\;e\;} \mathbf{M} \overset{\pi_L}{\underset{\pi_R}{\overset{\longrightarrow}{\underset{\longrightarrow}{\xrightarrow{c}}}}} \mathbf{M^2}$$

A monoid is then a particular **presheaf** on the above category C, *i.e.*, a functor

$$X \colon C^{\mathrm{op}} \to \mathbf{Set}$$

They are the ones such that

- $X(\mathbf{1})$ is a terminal set
- $(X(\mathbf{M^2}), X(\pi_L), X(\pi_R))$ is the product of $X(\mathbf{M})$ and $X(\mathbf{M})$
- the equations of monoids must hold: $X(c)(X(e)(x), y) = y$, *etc.*

These conditions can be expressed through **orthogonality conditions**.

Orthogonality

Let $\mathcal{C}$ be a category, $g\colon A \to B$ and $X \in \mathcal{C}$.

X is **orthogonal** to g when, for all $h\colon A \to X$, there is a unique $\overline{h}\colon B \to X$ such that $h = \overline{h} \circ g$.

$$\begin{tikzcd}
A \arrow[rr, "g"] \arrow[dr, "\forall h"'] & & B \arrow[dl, dotted, "\exists ! \bar{h}"] \\
& X &
\end{tikzcd}$$

Orthogonality

Let $O^{\mathcal{C}} \subseteq \mathcal{C}_1$ be a chosen set of **orthogonality morphisms**.

Orthogonality

Let $O^{\mathcal{C}} \subseteq \mathcal{C}_1$ be a chosen set of **orthogonality morphisms**.

$\mathcal{C}^{\perp}$: full subcategory of objects of $\mathcal{C}$ orthogonal to the arrows of $O^{\mathcal{C}}$.

Orthogonality

Let $O^{\mathcal{C}} \subseteq \mathcal{C}_1$ be a chosen set of **orthogonality morphisms**.

$\mathcal{C}^{\perp}$: full subcategory of objects of $\mathcal{C}$ orthogonal to the arrows of $O^{\mathcal{C}}$.

There is then a canonical inclusion functor

$$J\colon \mathcal{C}^{\perp} \to \mathcal{C}.$$

Orthogonality

Let $O^{\mathcal{C}} \subseteq \mathcal{C}_1$ be a chosen set of **orthogonality morphisms**.

$\mathcal{C}^{\perp}$: full subcategory of objects of $\mathcal{C}$ orthogonal to the arrows of $O^{\mathcal{C}}$.

There is then a canonical inclusion functor

$$J\colon \mathcal{C}^{\perp} \to \mathcal{C}.$$

Proposition (Adámek, Rosický)

If $\mathcal{C}$ is loc. fin. presentable, the canonical inclusion functor $J\colon \mathcal{C}^{\perp} \to \mathcal{C}$ has a left adjoint L:

$$\begin{tikzcd} \mathcal{C} \arrow[r, bend left, "L"] \arrow[r, phantom, "\perp"] & \mathcal{C}^{\perp} \arrow[l, bend left, "J"] \end{tikzcd}$$

Orthogonality conditions

The restrictions on presheaves can be expressed as orthogonality conditions.

Orthogonality conditions

The restrictions on presheaves can be expressed as orthogonality conditions.

Example for monoids:

$$\begin{tikzcd} \mathbf{1} & \mathbf{M} \arrow[l, "e"'] \arrow[r, shift left=2, "\pi_R"] \arrow[r, "c" description] \arrow[r, shift right=2, "\pi_L"'] & \mathbf{M^2} \end{tikzcd}$$

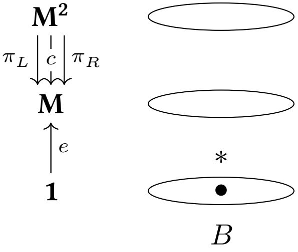
Orthogonality conditions

The restrictions on presheaves can be expressed as orthogonality conditions.

Example for monoids:

$$\begin{tikzcd} \mathbf{1} & \mathbf{M} \arrow[l, "e"'] \arrow[r, shift left=2, "\pi_R"] \arrow[r, "c" description] \arrow[r, shift right=2, "\pi_L"'] & \mathbf{M^2} \end{tikzcd}$$

Let B be the presheaf freely generated from one element $*$ in $B(\mathbf{1})$.



Orthogonality conditions

The restrictions on presheaves can be expressed as orthogonality conditions.

Example for monoids:

$$\begin{tikzcd} \mathbf{1} & \mathbf{M} \arrow[l, "e"'] \arrow[r, shift left=2, "\pi_R"] \arrow[r, "c" description] \arrow[r, shift right=2, "\pi_L"'] & \mathbf{M^2} \end{tikzcd}$$

Let B be the presheaf freely generated from one element $*$ in $B(\mathbf{1})$.

Let X in $\widehat{C}$. Then, $X(\mathbf{1})$ is a terminal set when X is orthogonal to $\emptyset \to B$

$$\begin{tikzcd} \emptyset \arrow[rr, "\emptyset"] \arrow[dr, "\forall H"'] & & B \arrow[dl, dotted, "\exists! \bar{H}"] \\ & X & \end{tikzcd}$$

Indeed, $\widehat{C}(B, X) \cong X(\mathbf{1})$, so that the condition says $X(\mathbf{1}) \cong \{*\}$.

Orthogonality conditions

The restrictions on presheaves can be expressed as orthogonality conditions.

Let

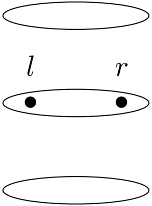
Orthogonality conditions

The restrictions on presheaves can be expressed as orthogonality conditions.

Let

- $A \in \widehat{C}$ freely gen. from two element l, r in $B(\mathbf{M})$

$$
\begin{tikzcd}
\mathbf{M}^2 \arrow[d, shift right=2, "\pi_L"'] \arrow[d, "c"] \arrow[d, shift left=2, "\pi_R"] \\
\mathbf{M} \\
\mathbf{1} \arrow[u, "e"']
\end{tikzcd}
$$



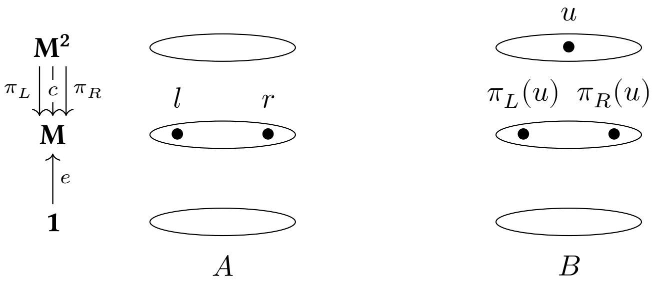
A

Orthogonality conditions

The restrictions on presheaves can be expressed as orthogonality conditions.

Let

- $A \in \widehat{C}$ freely gen. from two element l, r in $B(\mathbf{M})$
- $B \in \widehat{C}$ freely gen. from an element $u \in B(\mathbf{M^2})$

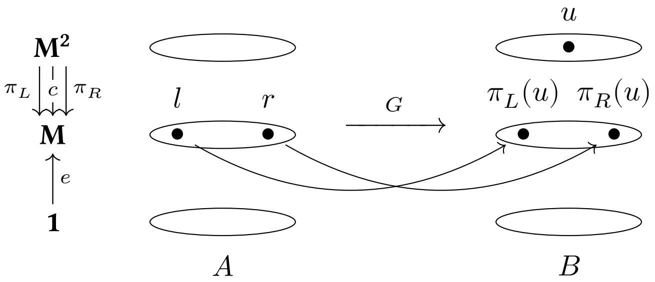


Orthogonality conditions

The restrictions on presheaves can be expressed as orthogonality conditions.

Let

- $A \in \widehat{C}$ freely gen. from two element l, r in $B(\mathbf{M})$
- $B \in \widehat{C}$ freely gen. from an element $u \in B(\mathbf{M^2})$
- $G\colon A \to B$ such that $G(l) = \pi_L(u)$ and $G(r) = \pi_R(u)$.



Orthogonality conditions

The restrictions on presheaves can be expressed as orthogonality conditions.

Let

- $A \in \widehat{C}$ freely gen. from two element l, r in $B(\mathbf{M})$
- $B \in \widehat{C}$ freely gen. from an element $u \in B(\mathbf{M^2})$
- $G\colon A \to B$ such that $G(l) = \pi_L(u)$ and $G(r) = \pi_R(u)$.

$(X(\mathbf{M^2}), X(\pi_L), X(\pi_R))$ is a product iif X is orthogonal to $G\colon A \to B$.

Indeed, $\widehat{C}(A, X) \cong X(\mathbf{M}) \times X(\mathbf{M})$ and $\widehat{C}(B, X) \cong X(\mathbf{M^2})$.

Orthogonality conditions

The restrictions on presheaves can be expressed as orthogonality conditions.

The equations of monoids can also be expressed as orthogonality conditions.

$$A^L \xrightarrow{G^L} B^L \qquad A^R \xrightarrow{G^R} B^R \qquad A^A \xrightarrow{G^A} B^A$$

Thus, $\mathbf{Mon} \simeq \widehat{C}^{\perp}$ for a set $O^C \subseteq \widehat{C}_1$ of orthogonality morphisms.

$$C \quad = \quad \begin{tikzcd} \mathbf{1} & \mathbf{M} \arrow[l, "e"'] \arrow[r, shift left=2, "\pi_L"] \arrow[r, "c" description] \arrow[r, shift right=2, "\pi_R"'] & \mathbf{M^2} \end{tikzcd}$$

Expressivity

With this representation, we can describe all **locally finitely presentable categories**.

Proposition

Every loc. fin. pres. category $\mathcal{C}$ can be described as

$$\mathcal{C} \simeq \widehat{C}^{\perp}$$

for some $C \in \mathbf{Cat}$ and $O^C \subseteq (\widehat{C})_1$.

L.f.p. categories are very common: $\mathbf{Set}$, $\mathbf{Mnd}$, $\mathbf{Cat}$, *etc.* ...

Encoding

(Sufficiently finite) l.f.p. can be encoded:

```
type path = Id of el | Cons of (el * path)

type fp_category = {
  objects   : set;
  arrows    : set;
  equations : (path * path) list
}

type lfp_category = {
  base_cat   : fp_category ;
  ortho_maps : ps_morph list
}
```

Outline

Compress information

$$\mathcal{F}: \quad \mathcal{C} \quad \to \quad \mathcal{D}$$

Goal: describe (some) functors between two l.f.p. categories $\mathcal{C}$ and $\mathcal{D}$.

We will need to filter some out.

Compress information

$$\mathcal{F}': \quad \widehat{C}^{\perp} \quad \to \quad \widehat{D}^{\perp}$$

First, we use the characterization: $\mathcal{C} \simeq \widehat{C}^{\perp}$ and $\mathcal{D} \simeq \widehat{D}^{\perp}$.

Compress information

$$\mathscr{F}'' \colon \quad \widehat{C} \quad \to \quad \widehat{D}^{\perp}$$

Then, let's actually define a functor $\mathscr{F}''$ on a larger domain.

In good cases, $\mathscr{F}'$ can then be recovered by precomposition with $J \colon \widehat{C}^{\perp} \to \widehat{C}$.

Compress information

$$\mathcal{F}''' : \quad \widehat{C} \quad \to \quad \widehat{D}$$

Also, let's actually define a functor $\mathcal{F}'''$ on a larger codomain.

In good cases, $\mathcal{F}''$ can be recovered by post-composition with $(-)^{\perp}$.

Compress information

$$\mathscr{F}'''' \colon \quad C \quad \to \quad \widehat{D}$$

Then, let's actually only define $\mathscr{F}''' \circ y$ where y is the Yoneda embedding

$$y \colon c \mapsto \mathrm{Hom}(-, c)$$

Compress information

$$\mathcal{F}'''' : \quad C \quad \to \quad \widehat{D}$$

If $\mathcal{F}'''$ is nice enough, it can be recovered using a **left Kan extension**:

$$\begin{tikzcd}
\widehat{C} \arrow[dr, "\mathcal{F}'''"] & \\
C \arrow[u, "y"] \arrow[r, "\mathcal{F}''''"'] \arrow[ur, phantom, "\Uparrow \alpha"] & \widehat{D}
\end{tikzcd}$$

Compress information

$$\mathcal{F}'''' : \quad C \quad \to \quad \widehat{D}$$

Under some finiteness hypothesis on C, D and $\mathcal{F}''''$, the latter can be described computationally.

Kan model

Conversely: one can start with a functor, called **Kan model**,

$$F\colon C \to \widehat{D}$$

and recover a functor $\mathcal{C} \to \mathcal{D}$.

Kan model

$$F\colon C \to \widehat{D}$$

Kan model

$$F' : \widehat{C} \to \widehat{D}$$

computed with a left Kan extension

$$\begin{tikzcd}
\widehat{C} \arrow[dr, "F'"] & \\
C \arrow[u, "y"] \arrow[r, "F"'] \arrow[ur, phantom, "\Uparrow \alpha"] & \widehat{D}
\end{tikzcd}$$

Kan model

$$\tilde{F} \colon \widehat{C} \to \widehat{D}^{\perp}$$

with $\tilde{F} = L \circ F'$

Kan model

$$\tilde{F}' : \widehat{C}^{\perp} \to \widehat{D}^{\perp}$$

with $\tilde{F}' = \tilde{F} \circ J$

Kan model

$$\bar{F}\colon \mathcal{C} \to \mathcal{D}$$

with $\bar{F} = \mathcal{C} \simeq \widehat{C}^{\perp} \xrightarrow{\tilde{F}'} \widehat{D}^{\perp} \simeq \mathcal{D}$

Kan model

Summary:

$$
\begin{tikzcd}
\mathcal{C} \arrow[d, "\simeq"'] \arrow[ddrrr, "\bar{F}"] & & & \\
\widehat{C}^{\perp} \arrow[d, "J"'] \arrow[drr, "\tilde{F}'"] & & & \\
\widehat{C} \arrow[r, "F'"] & \widehat{D} \arrow[r, "L"'] & \widehat{D}^{\perp} \arrow[r, "\simeq"'] & \mathcal{D} \\
C \arrow[u, "y"] \arrow[ur, "F"'] & & &
\end{tikzcd}
$$

Encoding Kan models

Assuming encodings for $\mathcal{C}$ and $\mathcal{D}$, Kan models $C \to \widehat{D}$ with finite images can be encoded.

```
type kan_model = {
  km_obj_map = el -> presheaf ;
  km_arr_map = el -> ps_morph
}
```

Kan extensions

What is actually a Kan extension doing?

Some intuition with a particular case but essential for the following.

Kan extensions

$$\begin{tikzcd} \widehat{C} \\ C \arrow[u, "y"] \arrow[r, "F"'] & \widehat{D} \end{tikzcd}$$

Given $F\colon C \to \widehat{D}$ and $y\colon C \to \widehat{C}$ the Yoneda embedding,

Kan extensions

$$\begin{tikzcd} \widehat{C} \arrow[drr, "F'"] & & \\ C \arrow[u, "y"] \arrow[rr, "F"'] \arrow[ur, phantom, "\Uparrow \alpha"] & & \widehat{D} \end{tikzcd}$$

a left Kan extension of F along y is a pair (F', α) which is universal in some sense.

Kan extensions

$$\begin{tikzcd}
\widehat{C} \arrow[dr, "F'"] & \\
C \arrow[u, "y"] \arrow[r, "F"'] \arrow[ur, phantom, "\Uparrow \alpha"] & \widehat{D}
\end{tikzcd}$$

Concretely:

$$F'(X) = \int^{c \in C} F(c) \otimes X(c)$$

Idea: for each $x \in X(c)$, there is one copy of $F(c)$ in $F'(X)$, adequately glued to other copies.

Kan extensions

$$\begin{tikzcd}
\widehat{C} \arrow[dr, "F'"] & \\
C \arrow[u, "y"] \arrow[r, "F"'] \arrow[ur, phantom, "\Uparrow \alpha"] & \widehat{D}
\end{tikzcd}$$

Even more concretely:

$$\coprod_{f\colon c \to c'} F(c) \otimes X(c') \overset{[\iota_{c'} \circ F(f) \otimes X(c')]_f}{\underset{[\iota_c \circ F(c) \otimes X(f)]_f}{\rightrightarrows}} \coprod_c F(c) \otimes X(c) \dashrightarrow \int^{c \in C} F(c) \otimes X(c) = F'(X)$$

which can be computed when F is encoded and X is finite!

Examples of functor descriptions

Taking

- $\mathbf{Set} \simeq \hat{1}^{\perp}$ with $O^{\mathbf{Set}} = \emptyset$
- $\mathbf{Set} \times \mathbf{Set} \simeq \widehat{1 \coprod 1}^{\perp}$ with $O^{\mathbf{Set} \times \mathbf{Set}} = \emptyset$

Examples of functor descriptions

Taking

- $\mathbf{Set} \simeq \hat{1}^{\perp}$ with $O^{\mathbf{Set}} = \emptyset$
- $\mathbf{Set} \times \mathbf{Set} \simeq \widehat{1 \coprod 1}^{\perp}$ with $O^{\mathbf{Set}\times\mathbf{Set}} = \emptyset$

the functor

$$\mathcal{F} : \qquad (X, Y) \in \mathbf{Set} \times \mathbf{Set} \qquad \mapsto \qquad X \in \mathbf{Set}$$

can be described by $\tilde{F} \colon 1 \coprod 1 \to \hat{1}$ where $\tilde{F}(0_L) = \{*\}$ and $\tilde{F}(0_R) = \emptyset$.

$$\begin{tikzcd}
\mathbf{Set} \times \mathbf{Set} \arrow[drr, "\mathcal{F}"] & & \\
1 \coprod 1 \arrow[u, "y"] \arrow[rr, "{[\{*\},\emptyset]}"'] \arrow[urr, phantom, "\Uparrow \alpha"] & & \mathbf{Set}
\end{tikzcd}$$

Examples of functor descriptions

Taking

- $\mathbf{Set} \simeq \hat{1}^{\perp}$ with $O^{\mathbf{Set}} = \emptyset$
- $\mathbf{Set} \times \mathbf{Set} \simeq \widehat{1 \coprod 1}^{\perp}$ with $O^{\mathbf{Set} \times \mathbf{Set}} = \emptyset$

the functor

$$\mathscr{F} : \qquad (X, Y) \in \mathbf{Set} \times \mathbf{Set} \qquad \mapsto \qquad X \in \mathbf{Set}$$

can be described by $\tilde{F} \colon 1 \coprod 1 \to \hat{1}$ where $\tilde{F}(0_L) = \{*\}$ and $\tilde{F}(0_R) = \emptyset$.

$$\begin{tikzcd}
\mathbf{Set} \times \mathbf{Set} \arrow[drr, "\mathscr{F}"] & & \\
1 \coprod 1 \arrow[u, "y"] \arrow[rr, "{[\{*\},\emptyset]}"'] \arrow[urr, phantom, "\Uparrow \alpha"] & & \mathbf{Set}
\end{tikzcd}$$

Idea: in $\mathbf{Set} \times \mathbf{Set}$, $0_L \rightsquigarrow (\{*\}, \emptyset)$, $0_R \rightsquigarrow (\emptyset, \{*\})$

Examples of functor descriptions

Taking

- $\mathbf{Set} \simeq \hat{1}^{\perp}$ with $O^{\mathbf{Set}} = \emptyset$
- $\mathbf{Mon} \simeq \widehat{C}^{\perp}$ with $O^{\mathbf{Mon}} = \{G^T, G^P, G^L, G^R, G^A\}$ and

$$C = \begin{tikzcd} \mathbf{1} & \mathbf{M} \arrow[l, "e"'] \arrow[r, shift left=2, "\pi_L"] \arrow[r, "c" description] \arrow[r, shift right=2, "\pi_R"'] & \mathbf{M^2} \end{tikzcd}$$

Examples of functor descriptions

Taking

- $\mathbf{Set} \simeq \hat{1}^{\perp}$ with $O^{\mathbf{Set}} = \emptyset$
- $\mathbf{Mon} \simeq \widehat{C}^{\perp}$ with $O^{\mathbf{Mon}} = \{G^T, G^P, G^L, G^R, G^A\}$ and

$$C = \begin{tikzcd} \mathbf{1} & \mathbf{M} \arrow[l, "e"'] \arrow[r, shift left=2, "\pi_L"] \arrow[r, "c" description] \arrow[r, shift right=2, "\pi_R"'] & \mathbf{M}^2 \end{tikzcd}$$

the free monoid functor

$$\mathcal{F}: \qquad S \in \mathbf{Set} \qquad \mapsto \qquad S^* \in \mathbf{Mon}$$

can be described by $\tilde{F}: 1 \to \widehat{C}$ where $\tilde{F}(0) = y(\mathbf{M})$.

$$\begin{tikzcd} \mathbf{Set} \arrow[dr] \arrow[drr, "\mathcal{F}"] & & \\ \mathbf{1} \arrow[u, "y"] \arrow[r, "y(\mathbf{M})"'] \arrow[r, phantom, shift left=4, "\Uparrow \alpha"] & \widehat{C} \arrow[r, "(-)^{\perp}"'] & \mathbf{Mon} \end{tikzcd}$$

Examples of functor descriptions

Taking

- $\mathbf{Set} \simeq \hat{1}^{\perp}$ with $O^{\mathbf{Set}} = \emptyset$
- $\mathbf{Mon} \simeq \widehat{C}^{\perp}$ with $O^{\mathbf{Mon}} = \{G^T, G^P, G^L, G^R, G^A\}$ and

$$C = \begin{tikzcd} \mathbf{1} & \mathbf{M} \arrow[l, "e"'] \arrow[r, shift left=2, "\pi_L"] \arrow[r, "c" description] \arrow[r, shift right=2, "\pi_R"'] & \mathbf{M^2} \end{tikzcd}$$

the free monoid functor

$$\mathcal{F}: \qquad S \in \mathbf{Set} \qquad \mapsto \qquad S^* \in \mathbf{Mon}$$

can be described by $\tilde{F}\colon 1 \to \widehat{C}$ where $\tilde{F}(0) = y(\mathbf{M})$.

Idea:

- in $\mathbf{Set}$, $0 \rightsquigarrow \{*\}$
- in $\mathbf{Mon}$, $y(\mathbf{M})$ corresponds to the free monoid $\{*\}^*$

Outline

Problem

Given a functor

$$\mathcal{F} : \mathcal{C} \to \mathcal{D}$$

described by a functor

$$\tilde{F} : C \to \widehat{D}$$

how can we check that $\mathcal{F}$ is a left adjoint?

Adjointness criterion

Proposition (Adámek, Rosický)

A functor $\mathcal{F} : \mathcal{C} \to \mathcal{D}$ between loc. fin. pres. cat. is a left adjoint if and only if it preserves all small colimits.

So: when is $\mathcal{F}$ preserving all small colimits?

Adjointness criterion

Assuming $\mathcal{C} \simeq \widehat{C}^{\perp}$ and $\mathcal{D} \simeq \widehat{D}^{\perp}$, and a Kan model $F\colon C \to \widehat{D}$,

Theorem

If the functor $\tilde{F}\colon \widehat{C} \to \widehat{D}^{\perp}$ sends the elements of O^{C} to isomorphisms, then $\bar{F}\colon \mathcal{C} \to \mathcal{D}$ preserves all colimits (and thus is a left adjoint).

Adjointness criterion

Assuming $\mathcal{C} \simeq \widehat{C}^{\perp}$ and $\mathcal{D} \simeq \widehat{D}^{\perp}$, and a Kan model $F\colon C \to \widehat{D}$,

Theorem

If the functor $\tilde{F}\colon \widehat{C} \to \widehat{D}^{\perp}$ sends the elements of O^{C} to isomorphisms, then $\bar{F}\colon \mathcal{C} \to \mathcal{D}$ preserves all colimits (and thus is a left adjoint).

When $\mathcal{C}, \mathcal{D}$ and F are encoded, checking the above property can be **mechanised**, if not **automatically computed**.

Indeed, checking that a morphism $G\colon A \to B \in \widehat{D}$ is sent to an isomorphism by L can be done by **playing a game**.

Outline

Product functors

Product functors can be given as inputs to the criterion:

Proposition

Given $\mathcal{C} \simeq \widehat{C}^{\perp}$ and $B \in \mathcal{C}$, the functor

$$X \mapsto X \times B$$

can be expressed by the Kan model $F\colon C \to \widehat{C}$, $c \mapsto A \times \mathrm{y}(c)$.

Indeed, working directly with $X, B \in \widehat{C}$, we have

$$X \times B \cong (\int^{c} \mathrm{y}(c) \otimes X(c)) \times B \cong \int^{c} (\mathrm{y}(c) \times B) \otimes X(c)$$

Cartesian closure

To show that a category $\mathcal{C}$ is cartesian closed, it is enough to show that all the functors $- \times B$ are left adjoint.

We can use our criterion to show that $- \times B$ is a left adjoint for a specific B.

Problem: infinite number of instances to check!

Cartesian closure

But, as presheaves

$$
\begin{aligned}
(-) \times B &\cong (-) \times \int^{c} \mathrm{y}(c) \otimes B(c) \\
&\cong \int^{c} ((-) \times \mathrm{y}(c)) \otimes B(c)
\end{aligned}
$$

Taking into account reflection,

Theorem

Given $\mathcal{C} \simeq \widehat{C}^{\perp}$, if the functors

$$L((-) \times \mathrm{y}(c))$$

are left adjoint for every $c \in C$, then $\mathcal{C}$ is cartesian closed.

Moreover, $L((-) \times \mathrm{y}(c))$ is modeled by the Kan model $d \mapsto \mathrm{y}(d) \times \mathrm{y}(c)$, so our l.a. criterion applies.

Outline

Non-example

Consider the functor

$$\begin{array}{rccc} \mathscr{F}: & \mathbf{Set} \times \mathbf{Set} & \to & \mathbf{Set} \\ & (X, Y) & \mapsto & X \times Y \end{array}$$

It is not a left adjoint. Let's see where the criterion fails.

Non-example

Consider the functor

$$\begin{array}{cccc} \mathscr{F}: & \mathbf{Set} \times \mathbf{Set} & \to & \mathbf{Set} \\ & (X, Y) & \mapsto & X \times Y \end{array}$$

It is not a left adjoint. Let's see where the criterion fails.

First, let's get a description for $\mathscr{F}$:

- $\mathbf{Set} \simeq \hat{\mathbf{1}}$
- $\mathbf{Set} \times \mathbf{Set} \simeq \widehat{\mathbf{1} \coprod \mathbf{1}}$

Non-example

Consider the functor

$$\begin{array}{rccc} \mathscr{F}\colon & \mathbf{Set}\times\mathbf{Set} & \to & \mathbf{Set} \\ & (X,Y) & \mapsto & X\times Y \end{array}$$

It is not a left adjoint. Let's see where the criterion fails.

First, let's get a description for $\mathscr{F}$:

- $\mathbf{Set} \simeq \hat{\mathbf{1}}$
- $\mathbf{Set}\times\mathbf{Set} \simeq \widehat{\mathbf{1}\coprod\mathbf{1}}$

But, $\mathscr{F}$ cannot be expressed by $\tilde{F}\colon \mathbf{1}\coprod\mathbf{1} \to \hat{\mathbf{1}}$.

Non-example

Consider the functor

$$\begin{array}{rccc} \mathscr{F}: & \mathbf{Set} \times \mathbf{Set} & \to & \mathbf{Set} \\ & (X, Y) & \mapsto & X \times Y \end{array}$$

It is not a left adjoint. Let's see where the criterion fails.

First, let's get a description for $\mathscr{F}$:

- $\mathbf{Set} \simeq \hat{\mathbf{1}}$
- $\mathbf{Set} \times \mathbf{Set} \simeq \widehat{\mathbf{1} \coprod \mathbf{1}}$

But, $\mathscr{F}$ cannot be expressed by $\tilde{F} \colon \mathbf{1} \coprod \mathbf{1} \to \hat{\mathbf{1}}$.

Indeed,

- $0_L \rightsquigarrow (\{*\}, \emptyset)$, $\qquad 0_R \rightsquigarrow (\emptyset, \{*\})$
- $(\{*\}, \emptyset)$ and $(\emptyset, \{*\})$ are mapped to $\emptyset$ by $\mathscr{F}$.
- but $\tilde{F} = \emptyset$ describes the functor $(X, Y) \mapsto \emptyset$.

Non-example

Another try: we add a (useless) product in the description of $\mathbf{Set} \times \mathbf{Set}$

- $\mathbf{Set} \simeq \hat{\mathbf{1}}$
- $\mathbf{Set} \times \mathbf{Set} \simeq \widehat{C}^{\perp}$

where

$$C = \begin{tikzcd} & p & \\ 0_L \arrow[ur, "\pi_L"] & & 0_R \arrow[ul, "\pi_R"'] \end{tikzcd}$$

Idea: $\quad 0_L \rightsquigarrow (\{*\}, \emptyset), \qquad 0_R \rightsquigarrow (\emptyset, \{*\}), \qquad p \rightsquigarrow (\{*\}, \{*\})$

Non-example

Another try: we add a (useless) product in the description of $\mathbf{Set} \times \mathbf{Set}$

- $\mathbf{Set} \simeq \hat{\mathbf{1}}$
- $\mathbf{Set} \times \mathbf{Set} \simeq \widehat{C}^{\perp}$

where

$$C = \begin{tikzcd} & p & \\ 0_L \arrow[ur, "\pi_L"] & & 0_R \arrow[ul, "\pi_R"'] \end{tikzcd}$$

and where we require orthogonality to $G\colon A \to B$:

$$\begin{tikzcd} & \bigcirc \arrow[dl, "\pi_L"'] \arrow[dr, "\pi_R"] & \\ \{*\} & & \{*\} \end{tikzcd} \quad \overset{G}{\longmapsto} \quad \begin{tikzcd} & \{*\} \arrow[dl, "\pi_L"'] \arrow[dr, "\pi_R"] & \\ \{*\} & & \{*\} \end{tikzcd}$$

i.e., given $X \in \widehat{C}^{\perp}$, $X(p)$ must be the product of $X(0_L)$ and $X(0_R)$.

Non-example

Another try: we add a (useless) product in the description of $\mathbf{Set} \times \mathbf{Set}$

- $\mathbf{Set} \simeq \hat{\mathbf{1}}$
- $\mathbf{Set} \times \mathbf{Set} \simeq \widehat{C}^{\perp}$

where

$$C = \begin{tikzcd} & p & \\ 0_L \arrow[ur, "\pi_L"] & & 0_R \arrow[ul, "\pi_R"'] \end{tikzcd}$$

Now, we can describe $\mathcal{F} : (X, Y) \mapsto X \times Y$ with

$$\begin{array}{llll} \tilde{F} : & C & \to & \hat{\mathbf{1}} \\ & 0_L & \mapsto & \emptyset \\ & 0_R & \mapsto & \emptyset \\ & p & \mapsto & \{*\} \end{array}$$

Non-example

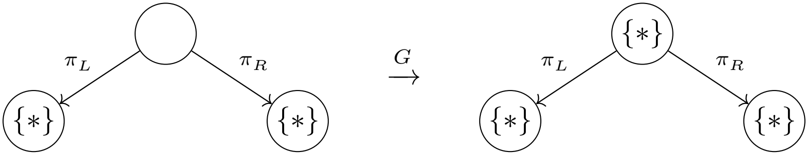
$\mathcal{F}\colon (X,Y)\mapsto X\times Y$ is not a left adjoint (coproducts are not preserved), so the criterion should not be satisfied.

We thus check that $(-)^{\perp}\circ\bar{F}\colon\widehat{C}\to\widehat{D}^{\perp}$ does not map $G\colon A\to B$ to an isomorphism.

Non-example

$\mathcal{F}\colon (X,Y) \mapsto X \times Y$ is not a left adjoint (coproducts are not preserved), so the criterion should not be satisfied.

We thus check that $(-)^{\perp} \circ \bar{F}\colon \widehat{C} \to \widehat{D}^{\perp}$ does not map $G\colon A \to B$ to an isomorphism.



Non-example

$\mathcal{F}\colon (X,Y) \mapsto X \times Y$ is not a left adjoint (coproducts are not preserved), so the criterion should not be satisfied.

We thus check that $(-)^{\perp} \circ \bar{F}\colon \widehat{C} \to \widehat{D}^{\perp}$ does not map $G\colon A \to B$ to an isomorphism.

$$\begin{tikzcd}
 & \bar{F}(A) \arrow[rrr, "\bar{F}(G)"] & & & \bar{F}(B) & \\
 & & & & \{*\} \arrow[u, dashed] & \\
\emptyset \arrow[uur, dashed] & & \emptyset \arrow[uul, dashed] & \emptyset \arrow[ur] & & \emptyset \arrow[ul]
\end{tikzcd}$$

Non-example

$\mathcal{F}\colon (X,Y) \mapsto X \times Y$ is not a left adjoint (coproducts are not preserved), so the criterion should not be satisfied.

We thus check that $(-)^{\perp} \circ \bar{F}\colon \widehat{C} \to \widehat{D}^{\perp}$ does not map $G\colon A \to B$ to an isomorphism.

$$\emptyset \xrightarrow{\quad \bar{F}(G) \quad} \{*\}$$

A bigger example

Let's show that this functor is a left adjoint:

$$\begin{aligned} \mathscr{F}: \quad \mathbf{Cat} \quad &\to \quad \mathbf{Set} \\ D \quad &\mapsto \quad D_0 \end{aligned}$$

A bigger example

Let's show that this functor is a left adjoint:

$$\begin{array}{rccc} \mathscr{F}: & \mathbf{Cat} & \to & \mathbf{Set} \\ & D & \mapsto & D_0 \end{array}$$

Consider the presentations of $\mathbf{Cat} \simeq \widehat{C}^{\perp}$ and $\mathbf{Set} \simeq \hat{1}$ with

$$C = \begin{tikzcd} \mathbf{C_0} \arrow[r, shift left=2, "\partial^+"] \arrow[r, shift right=2, "\partial^-"'] & \mathbf{C_1} \arrow[l, "\mathrm{id}" description] \arrow[r, shift left=2, "\pi_L"] \arrow[r, "c" description] \arrow[r, shift right=2, "\pi_R"'] & \mathbf{C_1^2} \end{tikzcd}$$

A bigger example

Let's show that this functor is a left adjoint:

$$\begin{array}{rccc} \mathcal{F}\colon & \mathbf{Cat} & \to & \mathbf{Set} \\ & D & \mapsto & D_0 \end{array}$$

Consider the presentations of $\mathbf{Cat} \simeq \widehat{C}^{\perp}$ and $\mathbf{Set} \simeq \hat{1}$ with

$$C = \quad \begin{tikzcd} \mathbf{C_0} \arrow[r, shift left=4, "\partial^+"] \arrow[r, shift right=4, "\partial^-"'] & \mathbf{C_1} \arrow[l, "\mathrm{id}" description] \arrow[r, shift left=4, "\pi_L"] \arrow[r, "c" description] \arrow[r, shift right=4, "\pi_R"'] & \mathbf{C_1^2} \end{tikzcd}$$

Consider the functor $\tilde{F}\colon C \to \mathbf{Set}$ where

$$\begin{array}{rcc} \tilde{F}(\mathbf{C_0}) & = & \{*\} \\ \tilde{F}(\mathbf{C_1}) & = & \{*_0, *_1\} \\ \tilde{F}(\mathbf{C_1^2}) & = & \{*_0, *_1, *_2\} \end{array}$$

A bigger example

Let's show that this functor is a left adjoint:

$$\begin{array}{rccc} \mathcal{F}\colon & \mathbf{Cat} & \to & \mathbf{Set} \\ & D & \mapsto & D_0 \end{array}$$

Consider the presentations of $\mathbf{Cat} \simeq \widehat{C}^{\perp}$ and $\mathbf{Set} \simeq \hat{1}$ with

$$C = \quad \mathbf{C_0} \begin{array}{c} \xrightarrow{\partial^+} \\ \xleftarrow{\mathrm{id}} \\ \xrightarrow[\partial^-]{} \end{array} \mathbf{C_1} \begin{array}{c} \xrightarrow{\pi_L} \\ \xrightarrow{c} \\ \xrightarrow[\pi_R]{} \end{array} \mathbf{C_1^2}$$

Consider the functor $\tilde{F}\colon C \to \mathbf{Set}$ where

$$\begin{array}{rcc} \tilde{F}(\mathbf{C_0}) & = & \{*\} \\ \tilde{F}(\mathbf{C_1}) & = & \{*_0, *_1\} \\ \tilde{F}(\mathbf{C_1^2}) & = & \{*_0, *_1, *_2\} \end{array}$$

Proposition

The functor $\mathcal{F}$ is presented by $\tilde{F}$.

A bigger example

Let's show that this functor is a left adjoint:

$$\begin{array}{rccc} \mathscr{F}\colon & \mathbf{Cat} & \to & \mathbf{Set} \\ & D & \mapsto & D_0 \end{array}$$

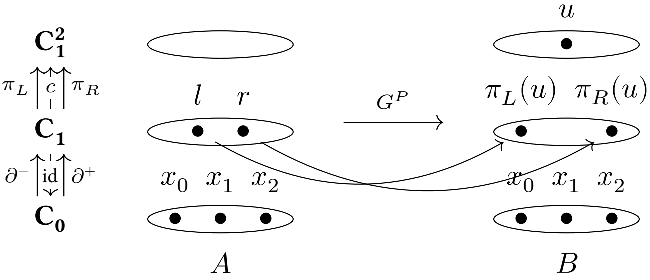
Let's compute whether $O^C = \{G^P, G^L, G^R, G^A\}$ is sent to isomorphisms by $\bar{F}\colon \widehat{C} \to \mathbf{Set}$

A bigger example

Let's show that this functor is a left adjoint:

$$\begin{array}{rccc} \mathcal{F}\colon & \mathbf{Cat} & \to & \mathbf{Set} \\ & D & \mapsto & D_0 \end{array}$$

Let's compute whether $O^C = \{G^P, G^L, G^R, G^A\}$ is sent to isomorphisms by $\bar{F}\colon \widehat{C} \to \mathbf{Set}$

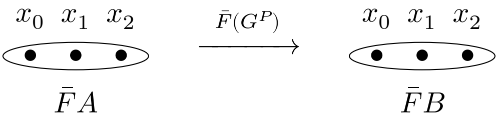


A bigger example

Let's show that this functor is a left adjoint:

$$\begin{array}{rccc} \mathcal{F}\colon & \mathbf{Cat} & \to & \mathbf{Set} \\ & D & \mapsto & D_0 \end{array}$$

Let's compute whether $O^C = \{G^P, G^L, G^R, G^A\}$ is sent to isomorphisms by $\bar{F}\colon \widehat{C} \to \mathbf{Set}$



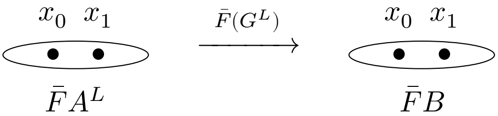
A bigger example

Let's show that this functor is a left adjoint:

$$\begin{array}{rccc} \mathcal{F}\colon & \mathbf{Cat} & \to & \mathbf{Set} \\ & D & \mapsto & D_0 \end{array}$$

Let's compute whether $O^C = \{G^P, G^L, G^R, G^A\}$ is sent to isomorphisms by $\bar{F}\colon \widehat{C} \to \mathbf{Set}$

Similarly, we have



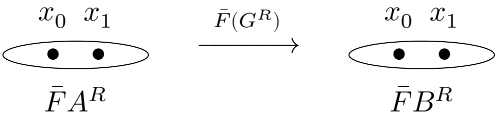
A bigger example

Let's show that this functor is a left adjoint:

$$\begin{array}{rccc} \mathcal{F}\colon & \mathbf{Cat} & \to & \mathbf{Set} \\ & D & \mapsto & D_0 \end{array}$$

Let's compute whether $O^C = \{G^P, G^L, G^R, G^A\}$ is sent to isomorphisms by $\bar{F}\colon \widehat{C} \to \mathbf{Set}$

Similarly, we have



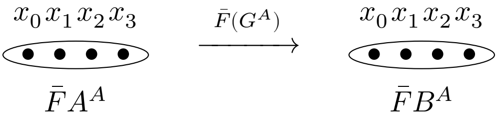
A bigger example

Let's show that this functor is a left adjoint:

$$\begin{array}{rccc} \mathcal{F}\colon & \mathbf{Cat} & \to & \mathbf{Set} \\ & D & \mapsto & D_0 \end{array}$$

Let's compute whether $O^C = \{G^P, G^L, G^R, G^A\}$ is sent to isomorphisms by $\bar{F}\colon \widehat{C} \to \mathbf{Set}$

Similarly, we have



A bigger example

Let's show that this functor is a left adjoint:

$$\begin{array}{rccc} \mathcal{F}\colon & \mathbf{Cat} & \to & \mathbf{Set} \\ & D & \mapsto & D_0 \end{array}$$

Let's compute whether $O^C = \{G^P, G^L, G^R, G^A\}$ is sent to isomorphisms by $\bar{F}\colon \widehat{C} \to \mathbf{Set}$

Proposition

The functor $\mathcal{F}$ is a left adjoint.

Example of product

We can use the criterion to show that $\mathbf{2} \times (-) \colon \mathbf{Cat} \to \mathbf{Cat}$ is a left adjoint where $\mathbf{Cat} \simeq \widehat{C}^{\perp}$ with

$$
C = \begin{tikzcd} \mathbf{C_0} \arrow[r, shift left=3, "\partial^+"] \arrow[r, shift right=3, "\partial^-"'] & \mathbf{C_1} \arrow[l, "\bar{\mathrm{id}}" description] \arrow[r, shift left=3, "\bar{\pi}_R"] \arrow[r, "\bar{c}" description] \arrow[r, shift right=3, "\bar{\pi}_L"'] & \mathbf{C_1^2} \end{tikzcd}
$$

Indeed, by computation, we check that every orthogonality morphism is sent to an isomorphism.

Outline

The reflection construction

Recall the adjunction

$$\begin{tikzcd} \widehat{D} \arrow[r, bend left, "(-)^{\perp}"] \arrow[r, phantom, "\perp"] & \widehat{D}^{\perp} \arrow[l, bend left, "J"] \end{tikzcd}$$

Given $H\colon X \to Y$, we have

$$\begin{tikzcd} X \arrow[r, "H"] \arrow[d, "\eta_X"'] & Y \arrow[d, "\eta_Y"] \\ JX^{\perp} \arrow[r, "JH^{\perp}"'] & JY^{\perp} \end{tikzcd}$$

The reflection construction

Recall the adjunction

$$\begin{tikzcd} \widehat{D} \arrow[r, bend left, "(-)^\perp"] \arrow[r, phantom, "\perp"] & \widehat{D}^\perp \arrow[l, bend left, "J"] \end{tikzcd}$$

Given $H\colon X \to Y$, we have

$$\begin{tikzcd} X \arrow[r, "H"] \arrow[d, "\eta_X"'] & Y \arrow[d, "\eta_Y"] \\ X^\perp \arrow[r, "H^\perp"'] & Y^\perp \end{tikzcd}$$

How to compute whether $H^\perp$ is an isomorphism?

The reflection construction

Recall the adjunction

$$\begin{tikzcd} \widehat{D} \arrow[r, bend left, "(-)^{\perp}"] \arrow[r, phantom, "\perp"] & \widehat{D}^{\perp} \arrow[l, bend left, "J"] \end{tikzcd}$$

Given $H\colon X \to Y$, we have

$$\begin{tikzcd} X \arrow[r, "H"] \arrow[d, "\eta_X"'] & Y \arrow[d, "\eta_Y"] \\ X^{\perp} \arrow[r, "H^{\perp}"'] & Y^{\perp} \end{tikzcd}$$

First: given $X \in \widehat{D}$, what is $\eta_X\colon X \to X^{\perp}$?

Idea: if X is not orthogonal, η_X is adding and merging the elements as required.

The reflection construction

Let $G\colon A \to B \in O^D$ be an orthogonality morphism.

The reflection construction

Let $G\colon A \to B \in O^D$ be an orthogonality morphism.

If some liftings are missing, as in

$$\begin{tikzcd} B \arrow[dr, dashed, "??"] & \\ & X \\ A \arrow[uu, "G"] \arrow[ur, "H"'] & \end{tikzcd}$$

we correct that using a pushout:

$$\begin{tikzcd} B \arrow[r, dashed] & X' \\ A \arrow[u, "G"] \arrow[r, "H"'] & X \arrow[u, dashed] \end{tikzcd}$$

The reflection construction

Let $G\colon A \to B \in O^D$ be an orthogonality morphism.

If some liftings are non-unique, as in

$$\begin{tikzcd} B \arrow[dr, shift left, "\bar{H}_1"] \arrow[dr, shift right, "\bar{H}_2"'] & \\ & X \\ A \arrow[uu, "G"] \arrow[ur, "H"'] & \end{tikzcd}$$

we correct that using a coequalizer:

$$\begin{tikzcd} B \arrow[r, shift left, "\bar{H}_1"] \arrow[r, shift right, "\bar{H}_2"'] & X \arrow[r, dashed] & X' \end{tikzcd}$$

The reflection construction

η_X is then the transfinite composition

$$X = X_0 \longrightarrow X_1 \longrightarrow X_2 \longrightarrow \cdots \longrightarrow X^{\perp}$$

The game

Given $H\colon X \to Y \in \widehat{D}$, how can we check that $H^{\perp}\colon X^{\perp} \to Y^{\perp}$ is an isomorphism?

Idea: progressively apply the moves of the reflection procedure until an isomorphism is obtained.

The game

$$H\colon X \to Y \in \widehat{D}$$

Four possible moves

The game

$$H\colon X \to Y \in \widehat{D}$$

Four possible moves

- add elements to X using a pushout with $G \in O^D$

$$H' : X' \to Y$$

The game

$$H\colon X \to Y \in \widehat{D}$$

Four possible moves

- add elements to X using a pushout with $G \in O^D$
- merge elements in X using a coequalizer of liftings of $G \in O^D$

$$H'\colon X' \to Y$$

The game

$$H\colon X \to Y \in \widehat{D}$$

Four possible moves

- add elements to X using a pushout with $G \in O^D$
- merge elements in X using a coequalizer of liftings of $G \in O^D$
- add elements to Y using a pushout with $G \in O^D$

$$H' : X \to Y'$$

The game

$$H\colon X \to Y \in \widehat{D}$$

Four possible moves

- add elements to X using a pushout with $G \in O^D$
- merge elements in X using a coequalizer of liftings of $G \in O^D$
- add elements to Y using a pushout with $G \in O^D$
- merge elements in Y using a coequalizer of liftings of $G \in O^D$

$$H' : X \to Y'$$

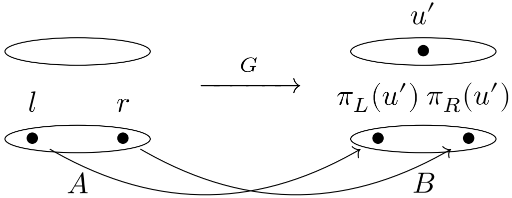
Play the game

Consider the category D where

$$D = \begin{tikzcd} e \\ b \arrow[u, shift left, "\pi_l"] \arrow[u, shift right, "\pi_r"'] \end{tikzcd}$$

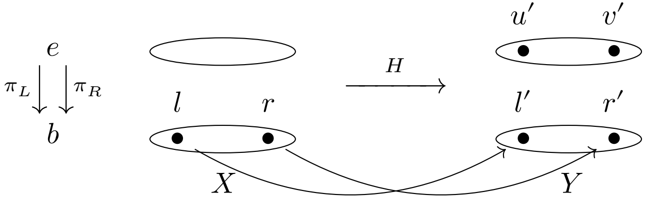
and with $O^D = \{G \colon A \to B\} \subseteq \widehat{D}$ with

$$\begin{tikzcd} e \arrow[d, shift right, "\pi_R"'] \arrow[d, shift left, "\pi_L"] \\ b \end{tikzcd}$$



Play the game

Show that $H\colon X \to Y \in \widehat{D}$ is sent to an isomorphism:

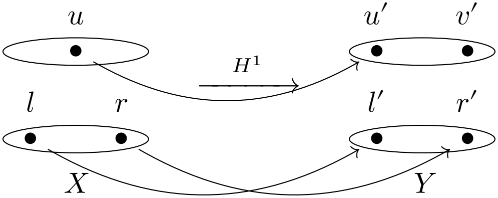


with $l' = \pi_l(u') = \pi_l(v')$ and $r' = \pi_r(u') = \pi_r(v')$

Play the game

Show that $H\colon X \to Y \in \widehat{D}$ is sent to an isomorphism:

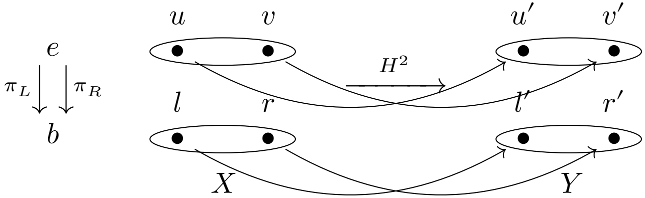
$$\begin{tikzcd} e \arrow[d, shift right, "\pi_L"'] \arrow[d, shift left, "\pi_R"] \\ b \end{tikzcd}$$



First, create a preimage for u'.

Play the game

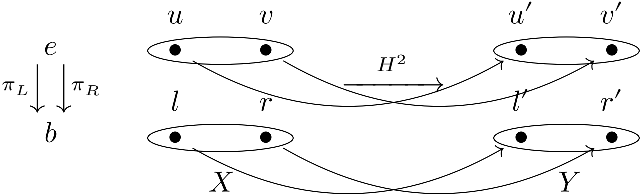
Show that $H\colon X \to Y \in \widehat{D}$ is sent to an isomorphism:



Then, create a preimage for v'.

Play the game

Show that $H\colon X \to Y \in \widehat{D}$ is sent to an isomorphism:

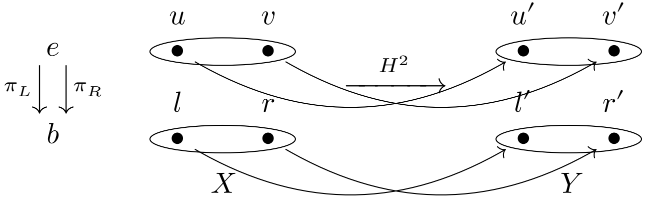


Then, create a preimage for v'.

We thus get an isomorphism.

Play the game

Show that $H\colon X \to Y \in \widehat{D}$ is sent to an isomorphism:



Then, create a preimage for v'.

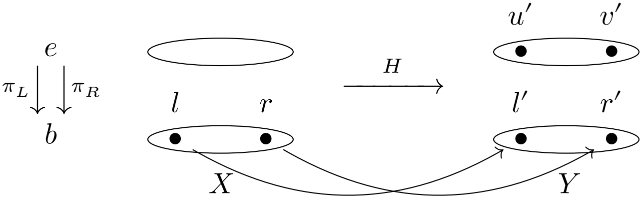
We used a "greedy strategy": add/merge when required and possible.

Proposition

The greedy strategy can decide whether $H^\perp$ is an isomorphism for finite $H\colon X \to Y \in \widehat{D}$.

Play the game

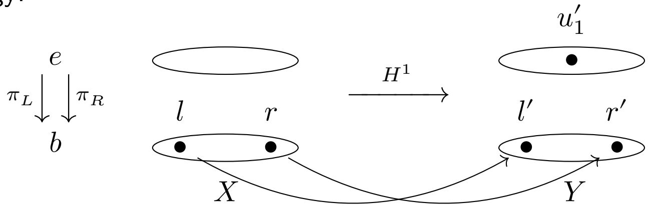
Another strategy:



with $l' = \pi_l(u') = \pi_l(v')$ and $r' = \pi_r(u') = \pi_r(v')$

Play the game

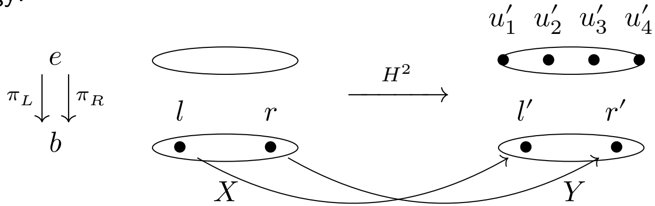
Another strategy:



First, merge u' and v', since they lift the same morphism.

Play the game

Another strategy:

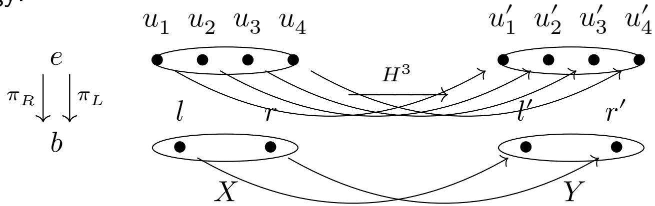


Then, create all the possible liftings in Y.

$$u'_1 = (l', r') \qquad u'_2 = (l', l') \qquad u'_3 = (r', r') \qquad u'_4 = (r', l')$$

Play the game

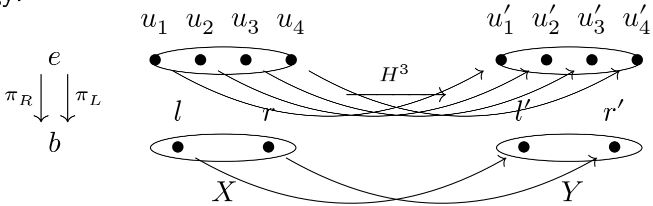
Another strategy:



Then, create all the possible liftings in X.

Play the game

Another strategy:

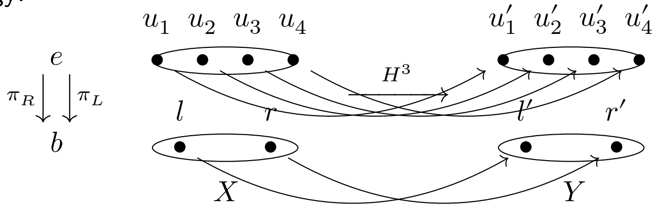


Then, create all the possible liftings in X.

We thus get an isomorphism.

Play the game

Another strategy:



Then, create all the possible liftings in X.

We used an "exhaustive strategy": add/merge whenever possible.

Proposition

The exhaustive strategy can decide whether $H^\perp$ is an isomorphism for finite $H\colon X \to Y \in \widehat{D}$.

Strategies in general

Winning the game can answer positively whether a morphism is sent to an isomorphism.

However,

- greedy strategies can be too stupid and miss some winnable games
- exhaustive strategies might not terminate

Future work: characterize the categories D and sets O^D for which these strategies terminate.

In any case: one can enter "manual mode" and provide a winning play.

Colimit preservation

Recall the definition of F:

$$\begin{tikzcd}
\widehat{C}^{\perp} \arrow[d, "J"'] \arrow[rrd, "F"] & & \\
\widehat{C} \arrow[r, "\bar{F}"] & \widehat{D} \arrow[r, "(-)^{\perp}"'] & \widehat{D}^{\perp} \\
C \arrow[u, "y"] \arrow[ru, "\tilde{F}"'] & &
\end{tikzcd}$$

Proposition

The functor $\bar{F} \colon \widehat{C} \to \widehat{D}$ preserves colimits.

Proof.

Indeed we have

$$\bar{F}(\operatorname{colim}_i X_i) \simeq \int^{c \in C_0} \tilde{F}(c) \otimes (\operatorname{colim}_i X_i)(c)$$

$\square$

Colimit preservation

Recall the definition of F:

$$
\begin{tikzcd}
\widehat{C}^{\perp} \arrow[d, "J"'] \arrow[drr, "F"] & & \\
\widehat{C} \arrow[r, "\bar{F}"] & \widehat{D} \arrow[r, "(-)^{\perp}"'] & \widehat{D}^{\perp} \\
C \arrow[u, "y"] \arrow[ur, "\tilde{F}"'] & &
\end{tikzcd}
$$

Proposition

The functor $\bar{F} \colon \widehat{C} \to \widehat{D}$ preserves colimits.

Proof.

Indeed we have

$$\bar{F}(\operatorname{colim}_i X_i) \simeq \int^{c \in C_0} \tilde{F}(c) \otimes \operatorname{colim}_i (X_i(c))$$

$\square$

Colimit preservation

Recall the definition of F:

$$
\begin{tikzcd}
\widehat{C}^{\perp} \arrow[d, "J"'] \arrow[drr, "F"] & & \\
\widehat{C} \arrow[r, "\bar{F}"] & \widehat{D} \arrow[r, "(-)^{\perp}"'] & \widehat{D}^{\perp} \\
C \arrow[u, "y"] \arrow[ur, "\tilde{F}"'] & &
\end{tikzcd}
$$

Proposition

The functor $\bar{F} \colon \widehat{C} \to \widehat{D}$ preserves colimits.

Proof.

Indeed we have

$$\bar{F}(\operatorname{colim}_i X_i) \simeq \int^{c \in C_0} \operatorname{colim}_i(\tilde{F}(c) \otimes X_i(c))$$

$\square$

Colimit preservation

Recall the definition of F:

$$\begin{tikzcd}
\widehat{C}^{\perp} \arrow[d, "J"'] \arrow[drr, "F"] & & \\
\widehat{C} \arrow[r, "\bar{F}"] & \widehat{D} \arrow[r, "(-)^{\perp}"'] & \widehat{D}^{\perp} \\
C \arrow[u, "y"] \arrow[ur, "\tilde{F}"'] & &
\end{tikzcd}$$

Proposition

The functor $\bar{F} \colon \widehat{C} \to \widehat{D}$ preserves colimits.

Proof.

Indeed we have

$$\bar{F}(\operatorname{colim}_i X_i) \simeq \operatorname{colim}_i\left(\int^{c \in C_0} \tilde{F}(c) \otimes X_i(c)\right)$$

$\square$

Colimit preservation

Recall the definition of F:

$$\begin{tikzcd}
\widehat{C}^{\perp} \arrow[d, "J"'] \arrow[drr, "F"] & & \\
\widehat{C} \arrow[r, "\bar{F}"] & \widehat{D} \arrow[r, "(-)^{\perp}"'] & \widehat{D}^{\perp} \\
C \arrow[u, "y"] \arrow[ur, "\tilde{F}"'] & &
\end{tikzcd}$$

Proposition

The functor $\bar{F}\colon \widehat{C} \to \widehat{D}$ preserves colimits.

Proof.

Indeed we have

$$\bar{F}(\operatorname{colim}_i X_i) \simeq \operatorname{colim}_i\left(\int^{c \in C_0} \tilde{F}(c) \otimes X_i(c)\right) \simeq \operatorname{colim}_i \bar{F}(X_i)$$

$\square$

Colimit preservation

$$\begin{tikzcd}
\widehat{C}^{\perp} \arrow[d, "J"'] \arrow[dr, "F"] & \\
\widehat{C} \arrow[r, "\bar{F}'"'] & \widehat{D}^{\perp}
\end{tikzcd}$$

Knowing that $\bar{F}' \mathrel{\widehat{=}} (-)^{\perp} \circ \bar{F}$ is preserving colimits, when F is?

Colimit preservation

$$\begin{tikzcd} \widehat{C}^{\perp} \arrow[d, "J"'] \arrow[dr, "F"] & \\ \widehat{C} \arrow[r, "\bar{F}'"'] & \widehat{D}^{\perp} \end{tikzcd}$$

Proposition (A-R)

The colimits in $\widehat{C}^{\perp}$ are the reflection of the ones computed in $\widehat{C}$:

$$\mathrm{colim}_i^{\widehat{C}^{\perp}} A_i \simeq (\mathrm{colim}_i^{\widehat{C}} J(A_i))^{\perp}$$

Thus, the unit of the reflection gives a canonical morphism

$$\eta\colon \mathrm{colim}_i^{\widehat{C}} JA_i \to J(\mathrm{colim}_i^{\widehat{C}^{\perp}} A_i)$$

Colimit preservation

$$\begin{tikzcd} \widehat{C}^{\perp} \arrow[d, "J"'] \arrow[dr, "F"] & \\ \widehat{C} \arrow[r, "\bar{F}'"'] & \widehat{D}^{\perp} \end{tikzcd}$$

Proposition (A-R)

The colimits in $\widehat{C}^{\perp}$ are the reflection of the ones computed in $\widehat{C}$:

$$\mathrm{colim}_i^{\widehat{C}^{\perp}} A_i \simeq (\mathrm{colim}_i^{\widehat{C}} J(A_i))^{\perp}$$

Thus, the unit of the reflection gives a canonical morphism

$$\bar{F}'\eta \colon \bar{F}'(\mathrm{colim}_i^{\widehat{C}} JA_i) \to \bar{F}'J(\mathrm{colim}_i^{\widehat{C}^{\perp}} A_i)$$

Colimit preservation

$$\begin{tikzcd}
\widehat{C}^{\perp} \arrow[d, "J"'] \arrow[dr, "F"] & \\
\widehat{C} \arrow[r, "\bar{F}'"'] & \widehat{D}^{\perp}
\end{tikzcd}$$

Proposition (A-R)

The colimits in $\widehat{C}^{\perp}$ are the reflection of the ones computed in $\widehat{C}$:

$$\mathrm{colim}_i^{\widehat{C}^{\perp}} A_i \simeq (\mathrm{colim}_i^{\widehat{C}} J(A_i))^{\perp}$$

Thus, the unit of the reflection gives a canonical morphism

$$\bar{F}'\eta \colon \bar{F}'(\mathrm{colim}_i^{\widehat{C}} JA_i) \to F(\mathrm{colim}_i^{\widehat{C}^{\perp}} A_i)$$

Colimit preservation

$$\begin{tikzcd} \widehat{C}^{\perp} \arrow[d, "J"'] \arrow[dr, "F"] & \\ \widehat{C} \arrow[r, "\bar{F}'"'] & \widehat{D}^{\perp} \end{tikzcd}$$

Proposition (A-R)

The colimits in $\widehat{C}^{\perp}$ are the reflection of the ones computed in $\widehat{C}$:

$$\operatorname{colim}_i^{\widehat{C}^{\perp}} A_i \simeq (\operatorname{colim}_i^{\widehat{C}} J(A_i))^{\perp}$$

Thus, the unit of the reflection gives a canonical morphism

$$\bar{F}'\eta\colon \operatorname{colim}_i^{\widehat{D}^{\perp}} (\bar{F}' J A_i) \to F(\operatorname{colim}_i^{\widehat{C}^{\perp}} A_i)$$

Colimit preservation

$$\begin{tikzcd} \widehat{C}^{\perp} \arrow[d, "J"'] \arrow[dr, "F"] & \\ \widehat{C} \arrow[r, "\bar{F}'"'] & \widehat{D}^{\perp} \end{tikzcd}$$

Proposition (A-R)

The colimits in $\widehat{C}^{\perp}$ are the reflection of the ones computed in $\widehat{C}$:

$$\mathrm{colim}_i^{\widehat{C}^{\perp}} A_i \simeq (\mathrm{colim}_i^{\widehat{C}} J(A_i))^{\perp}$$

Thus, the unit of the reflection gives a canonical morphism

$$\bar{F}'\eta\colon\ \mathrm{colim}_i^{\widehat{D}^{\perp}}(FA_i) \to F(\mathrm{colim}_i^{\widehat{C}^{\perp}} A_i)$$

Colimit preservation

$$\begin{tikzcd} \widehat{C}^{\perp} \arrow[d, "J"'] \arrow[dr, "F"] & \\ \widehat{C} \arrow[r, "\bar{F}'"'] & \widehat{D}^{\perp} \end{tikzcd}$$

Proposition

The functor $F\colon \widehat{C}^{\perp} \to \widehat{D}^{\perp}$ preserves colimits (and is a left adjoint) if and only if $\bar{F}'\eta_{\mathrm{colim}_i^{\widehat{C}} JA_i}$ is an isomorphism for all diagrams $i \mapsto A_i$ in $\widehat{C}^{\perp}$.

Colimit preservation

$$
\begin{tikzcd}
\widehat{C}^{\perp} \arrow[d, "J"'] \arrow[dr, "F"] & \\
\widehat{C} \arrow[r, "\bar{F}'"'] & \widehat{D}^{\perp}
\end{tikzcd}
$$

Proposition

The functor $F\colon \widehat{C}^{\perp} \to \widehat{D}^{\perp}$ *preserves colimits (and is a left adjoint) if and only if* $\bar{F}'\eta_{\mathrm{colim}_i^{\widehat{C}} JA_i}$ *is an isomorphism for all diagrams* $i \mapsto A_i$ *in* $\widehat{C}^{\perp}$.

Corollary

If $\bar{F}'\eta$ *is an isomorphism, then* F *preserves colimits (and is a left adjoint).*

Theorem

Suppose now that, for every orthogonality morphism $G \in O^C$, $\bar{F}(G)$ is an isomorphism.

Theorem

Suppose now that, for every orthogonality morphism $G \in O^C$, $\bar{F}(G)$ is an isomorphism.

If some liftings are missing for X, as in

$$\begin{tikzcd} B \arrow[dr, dashed, "??"] & \\ & X \\ A \arrow[uu, "G"] \arrow[ur, "H"'] & \end{tikzcd}$$

Theorem

Suppose now that, for every orthogonality morphism $G \in O^C$, $\bar{F}(G)$ is an isomorphism.

If some liftings are missing for X, as in

$$\begin{tikzcd}
B \arrow[dr, dashed, "??"] & \\
& X \\
A \arrow[uu, "G"] \arrow[ur, "H"'] &
\end{tikzcd}$$

we correct that using a pushout:

$$\begin{tikzcd}
B \arrow[r, dashed] & X' \\
A \arrow[u, "G"] \arrow[r, "H"'] & X \arrow[u, dashed]
\end{tikzcd}$$

Theorem

Suppose now that, for every orthogonality morphism $G \in O^C$, $\bar{F}(G)$ is an isomorphism.

If some liftings are missing for X, as in

$$\begin{tikzcd}
B \arrow[dr, dashed, "??"] & \\
 & X \\
A \arrow[uu, "G"] \arrow[ur, "H"'] &
\end{tikzcd}$$

...and we obtain the pushout

$$\begin{tikzcd}
\bar{F}B \arrow[r, dashed] & \bar{F}X' \\
\bar{F}A \arrow[u, "\bar{F}(G)"] \arrow[r, "\bar{F}(H)"'] & \bar{F}X \arrow[u, dashed]
\end{tikzcd}$$

where $\bar{F}(G)$ is an isomorphism. Thus, $\bar{F}X \simeq \bar{F}X'$.

Theorem

Suppose now that, for every orthogonality morphism G, $\bar{F}(G)$ is an isomorphism.

If liftings are non-unique, as in

$$\begin{tikzcd}
B \arrow[dr, shift left, "\bar{H}_1"] \arrow[dr, shift right, "\bar{H}_2"'] & \\
& X \\
A \arrow[uu, "G"] \arrow[ur, "H"'] &
\end{tikzcd}$$

Theorem

Suppose now that, for every orthogonality morphism G, $\bar{F}(G)$ is an isomorphism.

If liftings are non-unique, as in

$$\begin{tikzcd} B \arrow[drr, shift left, "\bar{H}_1"] \arrow[drr, shift right, "\bar{H}_2"'] & & \\ & & X \\ A \arrow[uu, "G"] \arrow[urr, "H"'] & & \end{tikzcd}$$

we correct that using a coequalizer:

$$\begin{tikzcd} B \arrow[r, shift left, "\bar{H}_1"] \arrow[r, shift right, "\bar{H}_2"'] & X \arrow[r, dashed] & X' \end{tikzcd}$$

Theorem

Suppose now that, for every orthogonality morphism G, $\bar{F}(G)$ is an isomorphism.

If liftings are non-unique, as in

$$\begin{tikzcd} B \arrow[dr, shift left, "\bar{H}_1"] \arrow[dr, shift right, "\bar{H}_2"'] & \\ & X \\ A \arrow[uu, "G"] \arrow[ur, "H"'] & \end{tikzcd}$$

...and we obtain the coequalizer:

$$\begin{tikzcd} \bar{F}B \arrow[r, shift left, "\bar{F}(\bar{H}_1)"] \arrow[r, shift right, "\bar{F}(\bar{H}_2)"'] & \bar{F}X \arrow[r, dashed] & \bar{F}X' \end{tikzcd}$$

with $\bar{F}(\bar{H}_1) \circ \bar{F}(G) = \bar{F}(\bar{H}_2) \circ \bar{F}(G)$, thus $\bar{F}(\bar{H}_1) = \bar{F}(\bar{H}_2)$ and $\bar{F}X \simeq \bar{F}X'$

Theorem

Thus, $\bar{F}\eta_X$ is a transfinite composition of isomorphism

$$\bar{F}X = \bar{F}X_0 \xrightarrow{\sim} \bar{F}X_1 \xrightarrow{\sim} \bar{F}X_2 \xrightarrow{\sim} \cdots \xrightarrow{\sim} \bar{F}X^{\perp}$$

Theorem

Thus, $\bar{F}\eta_X$ is a transfinite composition of isomorphism

$$\bar{F}X = \bar{F}X_0 \xrightarrow{\sim} \bar{F}X_1 \xrightarrow{\sim} \bar{F}X_2 \xrightarrow{\sim} \cdots \xrightarrow{\sim} \bar{F}X^{\perp}$$

Theorem

If, for all $G \in O^C$, $\bar{F}(G)$ is an isomorphism, then $\bar{F}\eta$ is an isomorphism.

Theorem

Thus, $\bar{F}\eta_X$ is a transfinite composition of isomorphism

$$\bar{F}X = \bar{F}X_0 \xrightarrow{\sim} \bar{F}X_1 \xrightarrow{\sim} \bar{F}X_2 \xrightarrow{\sim} \cdots \xrightarrow{\sim} \bar{F}X^{\perp}$$

Theorem
If, for all $G \in O^C$, $\bar{F}(G)$ is an isomorphism, then $\bar{F}\eta$ is an isomorphism.

Corollary
With the same hypothesis, F preserves colimits and is a left adjoint.

The end

Thank you!