

# Presenting Profunctors

NYC Category Theory Seminar

---

Joshua Meyers, Emilio Minichiello, Gabriel Goren Roig

May 22nd, 2024

- Going to talk about the paper "Presenting Profunctors" [MMR24], accepted for ACT 2024,
- In order to motivate this talk, need to talk about Categorical Database Theory.

**Idea** (Spivak):

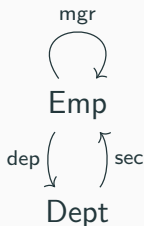
Database Schemas  $\longleftrightarrow$  Categories

Database Instances  $\longleftrightarrow$  Copresheaves

# Categorical Database Theory

**Ex:**

Let  $\mathcal{D}$  be the category



$$\text{mgr.mgr} = \text{mgr}$$

$$\text{sec.dep} = 1_{\text{Dept}}$$

# Categorical Database Theory

**Ex:**

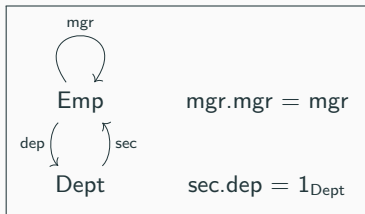
Let  $\mathcal{J}$  denote the functor  $\mathcal{J} : \mathcal{D} \rightarrow \mathbf{Set}$  given by

$$\mathcal{J}(\text{Emp}) = \{\text{Alice}, \text{Bob}, \text{Charlie}\}, \quad \mathcal{J}(\text{Dept}) = \{\text{CS}, \text{Math}\}$$

$$\mathcal{J}(\text{mgr}) = (\text{Alice} \mapsto \text{Alice}, \text{Bob} \mapsto \text{Bob}, \text{Charlie} \mapsto \text{Charlie})$$

$$\mathcal{J}(\text{dep}) = (\text{Alice} \mapsto \text{CS}, \text{Bob} \mapsto \text{Math}, \text{Charlie} \mapsto \text{Math}),$$

$$\mathcal{J}(\text{sec}) = (\text{CS} \mapsto \text{Alice}, \text{Math} \mapsto \text{Charlie})$$



# Categorical Database Theory

**Ex:**

Let  $\mathcal{J}$  denote the functor  $\mathcal{J} : \mathcal{D} \rightarrow \mathbf{Set}$  given by

$$\mathcal{J}(\text{Emp}) = \{\text{Alice}, \text{Bob}, \text{Charlie}\}, \mathcal{J}(\text{Dept}) = \{\text{CS}, \text{Math}\}$$

$$\mathcal{J}(\text{mgr}) = (\text{Alice} \mapsto \text{Alice}, \text{Bob} \mapsto \text{Bob}, \text{Charlie} \mapsto \text{Charlie})$$

$$\mathcal{J}(\text{dep}) = (\text{Alice} \mapsto \text{CS}, \text{Bob} \mapsto \text{Math}, \text{Charlie} \mapsto \text{Math}),$$

$$\mathcal{J}(\text{sec}) = (\text{CS} \mapsto \text{Alice}, \text{Math} \mapsto \text{Charlie})$$

Can visualize using tables

| Emp     | mgr     | dep  |
|---------|---------|------|
| Alice   | Alice   | CS   |
| Bob     | Charlie | Math |
| Charlie | Charlie | Math |

| Dept | sec     |
|------|---------|
| CS   | Alice   |
| Math | Charlie |

Now to **query** data in the categorical data model, we will use **profunctors**.

Recall that a profunctor  $\mathcal{P} : \mathcal{C} \nrightarrow \mathcal{D}$  is a functor  $\mathcal{P} : \mathcal{C}^{\text{op}} \times \mathcal{D} \rightarrow \mathbf{Set}$ .

Or equivalently a functor  $\mathcal{P} : \mathcal{C}^{\text{op}} \rightarrow \mathbf{Set}^{\mathcal{D}}$ .

Let  $\mathcal{P}$  be the query

”Select all employees who are their own manager.”

Let  $\mathcal{C}$  be the category with one object  $\text{Mgrs}$  and no non-identity morphisms.

We will construct a profunctor  $\mathcal{P} : \mathcal{C} \rightarrow \mathcal{D}$  that enacts the query above.



# Categorical Database Theory

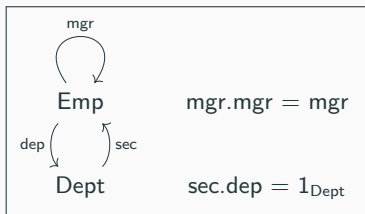
Let  $\mathcal{P} : \mathcal{C}^{\text{op}} \times \mathcal{D} \rightarrow \mathbf{Set}$  be defined as follows. Let  $\mathcal{P}_*$  denote the copresheaf  $\mathcal{P}(\text{Mgrs}) : \mathcal{D} \rightarrow \mathbf{Set}$ , defined as follows:

$$\mathcal{P}_*(\text{Emp}) = \{e, e.\text{dep}.\text{sec}\}, \quad \mathcal{P}_*(\text{Dept}) = \{e.\text{dep}\}$$

$$\mathcal{P}_*(\text{mgr}) = (e \mapsto e, e.\text{dep}.\text{sec} \mapsto e.\text{dep}.\text{sec})$$

$$\mathcal{P}_*(\text{dep}) = (e \mapsto e.\text{dep}, e.\text{dep}.\text{sec} \mapsto e.\text{dep})$$

$$\mathcal{P}_*(\text{sec}) = (e.\text{dep} \mapsto e.\text{dep}.\text{sec})$$



# Categorical Database Theory

Let  $\mathcal{P} : \mathcal{C}^{\text{op}} \times \mathcal{D} \rightarrow \mathbf{Set}$  be defined as follows. Let  $\mathcal{P}_*$  denote the copresheaf  $\mathcal{P}(\text{Mgrs}) : \mathcal{D} \rightarrow \mathbf{Set}$ , defined as follows:

$$\mathcal{P}_*(\text{Emp}) = \{e, e.\text{dep}.\text{sec}\}, \quad \mathcal{P}_*(\text{Dept}) = \{e.\text{dep}\}$$

$$\mathcal{P}_*(\text{mgr}) = (e \mapsto e, e.\text{dep}.\text{sec} \mapsto e.\text{dep}.\text{sec})$$

$$\mathcal{P}_*(\text{dep}) = (e \mapsto e.\text{dep}, e.\text{dep}.\text{sec} \mapsto e.\text{dep})$$

$$\mathcal{P}_*(\text{sec}) = (e.\text{dep} \mapsto e.\text{dep}.\text{sec})$$

| $\mathcal{P}_*(\text{Emp})$ | $\mathcal{P}_*(\text{mgr})$ | $\mathcal{P}_*(\text{dep})$ |
|-----------------------------|-----------------------------|-----------------------------|
| e                           | e                           | e.dep                       |
| e.dep.sec                   | e.dep.sec                   | e.dep                       |

| $\mathcal{P}_*(\text{Dept})$ | $\mathcal{P}_*(\text{sec})$ |
|------------------------------|-----------------------------|
| e.dep                        | e.dep.sec                   |

Given a profunctor  $\mathcal{P} : \mathcal{C} \rightarrow \mathcal{D}$ , we obtain a functor

$$\text{Eval}_{\mathcal{P}} : \mathbf{Set}^{\mathcal{D}} \rightarrow \mathbf{Set}^{\mathcal{C}}$$

defined objectwise for  $\mathcal{J} \in \mathbf{Set}^{\mathcal{D}}$  and  $c \in \mathcal{C}$  by

$$\text{Eval}_{\mathcal{P}}(\mathcal{J})(c) = \mathbf{Set}^{\mathcal{D}}(\mathcal{P}(c), \mathcal{J}).$$

We call this the **evaluation** of  $\mathcal{J}$  by  $\mathcal{P}$ .

# Categorical Database Theory

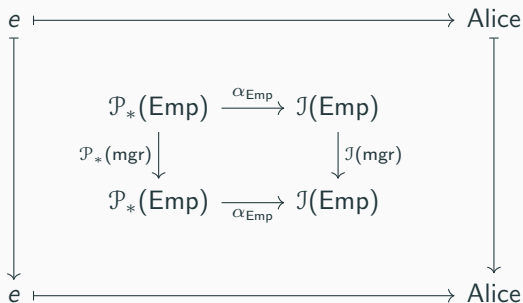
With our example from before, we want to compute  $\text{Eval}_{\mathcal{P}}(\mathcal{J})$ .

Now  $\text{Eval}_{\mathcal{P}}(\mathcal{J})(\text{Mgrs}) = \mathbf{Set}^{\mathcal{D}}(\mathcal{P}(\text{Mgrs}), \mathcal{J}) = \mathbf{Set}^{\mathcal{D}}(\mathcal{P}_*, \mathcal{J})$ , the set of natural transformations.

Now the function  $\alpha_{\text{Emp}} : \mathcal{P}_*(\text{Emp}) \rightarrow \mathcal{J}(\text{Emp})$ , defined by  $e \mapsto \text{Alice}$ ,  $e.\text{dep.sec} \mapsto \text{Alice}$  extends to a natural transformation. For example, the following diagram commutes

$$\begin{array}{ccc} \mathcal{P}_*(\text{Emp}) & \xrightarrow{\alpha_{\text{Emp}}} & \mathcal{J}(\text{Emp}) \\ \mathcal{P}_*(\text{mgr}) \downarrow & & \downarrow \mathcal{J}(\text{mgr}) \\ \mathcal{P}_*(\text{Emp}) & \xrightarrow{\alpha_{\text{Emp}}} & \mathcal{J}(\text{Emp}) \end{array}$$

# Categorical Database Theory

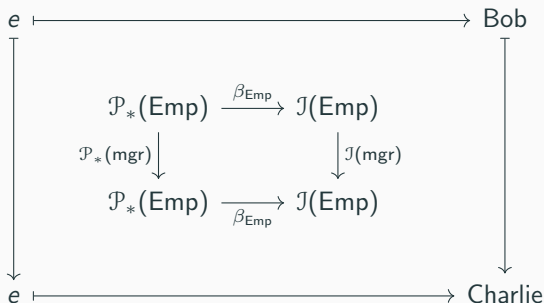


$$\mathcal{P}_*(\text{mgr}) = 1_{\mathcal{P}_*(\text{Emp})}$$

$$\mathcal{J}(\text{mgr}) = (\text{Alice} \mapsto \text{Alice}, \text{Bob} \mapsto \text{Charlie}, \text{Charlie} \mapsto \text{Charlie})$$

# Categorical Database Theory

However the function  $\beta_{\text{Emp}} : \mathcal{P}_*(\text{Emp}) \rightarrow \mathcal{J}(\text{Emp})$  that sends  $e \mapsto \text{Bob}$  and  $e.\text{dep.sec} \mapsto \text{Charlie}$  does not extend to a natural transformation



So we have

$$\text{Eval}_{\mathcal{P}}(\mathcal{J})(\text{Emp}) = \{(e \mapsto \text{Alice}), (e \mapsto \text{Charlie})\}$$

or as a table

| $\text{Eval}_{\mathcal{P}}(\mathcal{J})(\text{Emp})$ |
|--|
| Alice  |
| Charlie  |

This is the result of the query.

Thus we can extend the idea from before:

Database Schemas  $\longleftrightarrow$  Categories

Database Instances  $\longleftrightarrow$  Copresheaves

Database Queries  $\longleftrightarrow$  Profunctors



Further important property of profunctors: **They compose!**

Think of profunctors as a categorification of relations. A relation  $R \subseteq A \times B$  is equivalently a function  $R : A \times B \rightarrow \{0, 1\}$ .

We can compose relations  $R : A \rightarrow B$ ,  $S : B \rightarrow C$  to obtain a relation  $(R \odot S) : A \rightarrow C$  defined by

$$(R \odot S) = \{(a, c) \in A \times C \mid \exists b \in B, R(a, b) \text{ and } S(b, c)\}.$$

In a similar vein, given profunctors  $\mathcal{P} : \mathcal{C} \rightarrow \mathcal{D}$ ,  $Q : \mathcal{D} \rightarrow \mathcal{E}$ , we can obtain a composite profunctor  $(P \odot Q) : \mathcal{C} \rightarrow \mathcal{E}$  using the following coend formula

$$(P \odot Q)(c, e) = \int^{d \in \mathcal{D}} P(c, d) \times Q(d, e).$$

Furthermore if  $\mathcal{J} \in \mathbf{Set}^{\mathcal{E}}$ , then

$$\text{Eval}_{P \odot Q}(\mathcal{J}) \cong \text{Eval}_P(\text{Eval}_Q(\mathcal{J})).$$

This means that we can compose database queries!

However, if we want to use the categorical model on a computer, we must use **presentations**.

The main point of this paper:

*Presentations of profunctors are surprisingly subtle!*

Let us discuss presentations.

A **category signature**  $\Sigma$  consists of sets

- $\text{Sort}(\Sigma)$ , whose elements we call **sorts**,
- $\text{Fun}(\Sigma)$ , whose elements we call **function symbols**, and
- two functions  $s, t : \text{Fun}(\Sigma) \rightarrow \text{Sort}(\Sigma)$ , called the **source** and **target** functions. We write  $f : c \rightarrow c'$  to mean that  $s(f) = c$  and  $t(f) = c'$ .

A **path** in  $\Sigma$  consists of a (possibly empty) sequence of function symbols, which we write like  $p = f_0.f_1.\dots.f_n$ , and let  $1_c$  denote the empty path on a sort  $c$ . We write  $p : c \rightarrow c'$  if  $s(f_0) = c$  and  $t(f_n) = c'$ . Let  $\text{Path}(\Sigma)$  denote the set of paths in  $\Sigma$ .

# Category Presentations

A map  $F : \Sigma \rightarrow \Sigma'$  of category signatures consists of functions

- $F_0 : \text{Sort}(\Sigma) \rightarrow \text{Sort}(\Sigma')$ , and
- $F_1 : \text{Fun}(\Sigma) \rightarrow \text{Path}(\Sigma')$ ,

such that if  $f : c \rightarrow c'$  is in  $\text{Fun}(\Sigma)$ , then  $F_1(f)$  is a path with  $F_1(f) : F_0(c) \rightarrow F_0(c')$ . We often simply write  $F$  instead of  $F_0$  and  $F_1$ .

We extend  $F$  to a function  $F : \text{Path}(\Sigma) \rightarrow \text{Path}(\Sigma')$  by setting

$$F(f_0 \cdot \dots \cdot f_n) = F(f_0) \cdot \dots \cdot F(f_n), \quad \text{and} \quad F(1_c) = 1_{F(c)}$$

Given a category signature  $\Sigma$ , an **equation** in  $\Sigma$  consists of a pair  $(p, q)$  of paths  $p$  and  $q$  with  $s(p) = s(q) = c$  and  $t(p) = t(q) = c'$ . We write this as  $p = q : c \rightarrow c'$ .

A **category presentation**  $C$  consists of a pair  $C = (C_\Sigma, C_E)$  where  $C_\Sigma$  is a category signature and  $C_E$  is a set of equations over  $C_\Sigma$ . We write  $p =_C q$  to mean that  $(p, q) \in C_E$ .

# Category Presentations

Define the relation  $\approx_C$  on  $\text{Path}(C) := \text{Path}(C_\Sigma)$  by the following inference rules

$$\frac{p =_C q}{p \approx_C q} \quad \frac{}{p \approx_C p} \quad \frac{p \approx_C q \quad q \approx_C r}{p \approx_C r} \quad \frac{p \approx_C q}{q \approx_C p}$$

$$\frac{f : c' \rightarrow c'' \quad p \approx_C q : c \rightarrow c'}{p.f \approx_C q.f} \quad \frac{f : c \rightarrow c' \quad p \approx_C q : c' \rightarrow c''}{f.p \approx_C f.q}$$

where  $p, q$  are paths and  $f$  is a function symbol. We call  $\approx_C$  the **provable equality relation**.

# Category Presentations

A map  $F : C \rightarrow D$  of category presentations is a map  $F : C_\Sigma \rightarrow D_\Sigma$  of category signatures such that if  $p =_C q$ , then  $F(p) \approx_D F(q)$ . Let **CatPr** denote the category of category presentations.

Given a category presentation  $C$ , let  $\llbracket C \rrbracket$  denote the category with

- $\text{Obj}(\llbracket C \rrbracket) = \text{Sort}(C)$ , and
- $\text{Mor}(\llbracket C \rrbracket) = \text{Path}(C) / \approx_C$ .

We call  $\llbracket C \rrbracket$  the **category presented by**  $C$ , or simply its **semantics**.

This construction defines a functor  $\llbracket - \rrbracket : \mathbf{CatPr} \rightarrow \mathbf{Cat}$ .



# Category Presentations

If  $F, F' : C \rightarrow D$  are maps of cat. pres., then we write  $F \approx F'$  if  $F(f) \approx_D F'(f)$  for every function symbol  $f$  in  $C$ . We say that  $F$  and  $F'$  are **provably equal**.

Let  $\mathbf{CatPr}_{\approx}$  denote the category of cat. pres. with equivalence classes of provably equal maps of category presentations.

The induced functor  $(|-) : \mathbf{CatPr}_{\approx} \rightarrow \mathbf{Cat}$  is now an equivalence.

# Category Presentations

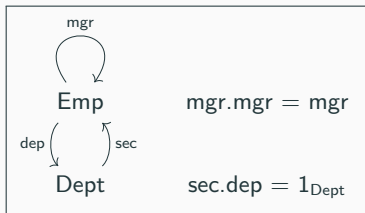
**Ex:**

Consider the category presentation  $C$  with

- $\text{Sort}(C) = \{\text{Emp}, \text{Dept}\}$ ,
- $\text{Fun}(C) = \{\text{mgr}, \text{dep}, \text{sec}\}$ ,
- $C_E = \{\text{mgr.mgr} = \text{mgr}, \text{sec.dep} = 1_{\text{Dept}}\}$

$(C)$  has morphisms

$[1_{\text{Emp}}], [1_{\text{Dept}}], [\text{mgr}], [\text{sec}], [\text{dep}], [\text{sec.mgr.dep}], [\text{mgr.dep}], \dots$



# Profunctor Presentations

How do we define profunctor presentations?

First, let us give the following construction. Given a profunctor  $\mathcal{P} : \mathcal{C} \rightarrow \mathcal{D}$ , let  $\tilde{\mathcal{P}}$  denote the category with

- $\text{Obj}(\tilde{\mathcal{P}}) = \text{Obj}(\mathcal{C}) + \text{Obj}(\mathcal{D})$ ,

- 

$$\tilde{\mathcal{P}}(x, y) = \begin{cases} \mathcal{C}(x, y) & x, y \in \mathcal{C} \\ \mathcal{P}(x, y) & x \in \mathcal{C}, y \in \mathcal{D} \\ \emptyset & x \in \mathcal{D}, y \in \mathcal{C} \\ \mathcal{D}(x, y) & x, y \in \mathcal{D}. \end{cases}$$

Composition of morphisms in  $\tilde{\mathcal{P}}$  is defined using the functoriality of  $\mathcal{P}$ .

We call  $\tilde{\mathcal{P}}$  the **collage** of  $\mathcal{P}$ .

# Profunctor Presentations

Think of  $\tilde{\mathcal{P}}$  as a bridge from  $\mathcal{C}$  to  $\mathcal{D}$ .

There is a functor  $\pi : \tilde{\mathcal{P}} \rightarrow 2$ , where  $2$  is the category  $0 \leq 1$ . The functor  $\pi$  sends all of  $\mathcal{C}$  to  $0$ , all of  $\mathcal{D}$  to  $1$  and all morphisms between to the unique morphism  $\leq$ .

In fact we have the following pullbacks in **Cat**

$$\begin{array}{ccccc} \mathcal{C} & \hookrightarrow & \tilde{\mathcal{P}} & \longleftarrow & \mathcal{D} \\ \downarrow & \lrcorner & \downarrow \pi & \lrcorner & \downarrow \\ * & \xrightarrow{\quad 0 \quad} & 2 & \xleftarrow{\quad 1 \quad} & * \end{array}$$

This construction defines a functor  $\widetilde{(-)} : \mathbf{Prof} \rightarrow \mathbf{Cat}/2$ .

**Proposition:** The functor  $\widetilde{(-)}$  is an equivalence of categories.

Thus we will use collages to present profunctors.

# Profunctor Presentations

Given category signatures  $\Sigma$  and  $\Sigma'$ , an **uncurried profunctor signature**  $\Pi$  consists of a set of  $\text{Fun}(\Pi)$ , whose elements are called **profunctor function symbols**, along with two functions  $s : \text{Fun}(\Pi) \rightarrow \text{Sort}(\Sigma)$  and  $t : \text{Fun}(\Pi) \rightarrow \text{Sort}(\Sigma')$ .

We define the associated category signature  $|\Pi|$  to have  
 $\text{Sort}(|\Pi|) = \text{Sort}(\Sigma) + \text{Sort}(\Sigma')$  and  
 $\text{Fun}(|\Pi|) = \text{Fun}(\Sigma) + \text{Fun}(\Pi) + \text{Fun}(\Sigma')$ .

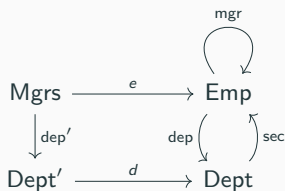
We call a path  $p : c \rightarrow d$  in  $|\Pi|$  a **cross-path** if  $c \in \Sigma$  and  $d \in \Sigma'$ . Let  $\text{CPath}(\Pi)$  denote the set of cross-paths in  $|\Pi|$ .

By an **uncurried profunctor equation** we mean an equation  $p = q : c \rightarrow d$  between cross-paths.

# Profunctor Presentations

Given category presentations  $C$  and  $D$ , a  $(C, D)$ -**uncurried profunctor presentation**  $P$  consists of a pair  $P = (P_\Sigma, P_E)$ , where  $P_\Sigma$  is an uncurried profunctor signature from  $C_\Sigma$  to  $D_\Sigma$ , and  $P_E$  is a set of uncurried profunctor equations.

**Ex:**



$$\text{mgr.mgr} = \text{mgr}$$

$$\text{sec.dep} = 1_{\text{Dept}}$$

$$e.\text{dep} = \text{dep}'.d$$

# Profunctor Presentations

If  $P = (P_\Sigma, P_E)$  is an uncurried profunctor presentation from  $C$  to  $D$ , we define the associated category presentation  $|P|$  by  $|P|_\Sigma = P_\Sigma$  and  $|P|_E = C_E + P_E + D_E$ . We say that  $p =_P q$  if  $(p, q) \in P_E$ , and we define  $p \approx_P q$  to be the restriction of  $\approx_{|P|}$  to cross-paths.

A morphism  $F : P \rightarrow P'$  of  $(C, D)$ -uncurried profunctor presentations is a function  $F : \text{Fun}(P_\Sigma) \rightarrow \text{CPath}(P'_\Sigma)$  such that the corresponding map  $|F| : |P| \rightarrow |P'|$  is a map of category presentations.

Let  $\mathbf{UnCurr}(C, D)$  denote the category of  $(C, D)$ -uncurried profunctor presentations.



We define  $\langle P \rangle$  in the obvious way, obtaining a functor  $\langle - \rangle : \mathbf{UnCurr}(C, D) \rightarrow \mathbf{Prof}(\langle C \rangle, \langle D \rangle)$ .

Suppose that  $C$  and  $D$  are finite category presentations (finitely many sorts, function symbols and equations). We say that a profunctor  $\mathcal{P} : \langle C \rangle \rightarrow \langle D \rangle$  is **finitely uncurried presentable** if there exists a finite uncurried profunctor presentation (finitely many profunctor function symbols)  $P$  such that  $\langle P \rangle \cong \mathcal{P}$ .

**Thm**[MMR]: The class of finitely uncurried presentable profunctors is not closed under composition.


Proof: Let

- $C$  be the cat. pres. with one sort  $c$  and no function symbols,
- $D$  be the cat. pres. with one sort  $d$  and one function symbol  $f : d \rightarrow d$  and no equations,
- $E$  be the cat. pres. with one sort  $e$  and no function symbols
- $P : C \leftrightarrow D$  be the uncurried prof. pres. with one symbol  $p : c \rightarrow d$  and no equations,
- $Q : D \leftrightarrow E$  be the uncurr. prof. pres. with one symbol  $q : d \rightarrow e$  and no equations.

# Profunctor Presentations

**Thm**[MMR]: The class of finitely uncurried presentable profunctors is not closed under composition.

Proof:

$$c \xrightarrow{p} d \xrightarrow{q} e$$


Then  $((P) \odot (Q))(c, e) = \{[p.f^n.q] : n \geq 0\}$ . Let  $R$  be an uncurried profunctor presentation from  $C$  to  $E$  such that  $(R) \cong ((P) \odot (Q))$ . Then  $R$  must have infinitely many profunctor function symbols, since there are no function symbols in  $C$  or  $E$ . Thus  $((P) \odot (Q))$  is not finitely uncurried presentable.

This is a major defect of uncurried profunctor presentations.

Let us introduce a different notion of profunctor presentation. First let us define instance presentations.

Given a category  $\mathcal{C}$ , an **instance** on  $\mathcal{C}$  is just a copresheaf  $\mathcal{J} : \mathcal{C} \rightarrow \mathbf{Set}$ .

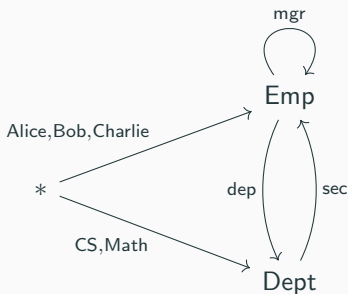
Note that an instance is the same thing as a profunctor  $\mathcal{J} : \mathbf{1} \rightarrow \mathcal{C}$ , where  $\mathbf{1}$  is the terminal category.

# Profunctor Presentations

Let  $1$  denote the category presentation with one sort  $*$ , with no function symbols or equations. Given a category presentation  $C$ , an **instance presentation** on  $C$  is a  $(1, C)$ -uncurried profunctor presentation. In other words  $C\text{Inst} := \mathbf{UnCurr}(1, C)$ .

If  $I$  is an instance presentation on a cat. pres.  $C$ , we call the cross-paths  $x : * \rightarrow c$  the **generators** of  $I$ .

**Ex:**



$$\text{mgr.mgr} = \text{mgr}$$

$$\text{sec.dep} = 1_{\text{Dept}}$$

$$\text{Alice.mgr} = \text{Alice} \quad \dots$$

Now given category presentations  $C$  and  $D$ , a **curried profunctor presentation**  $P : C \rightarrow D$  consists of an assignment of a  $D$ -instance presentation  $P(c)$  to every sort  $c$  of  $C$ , along with a morphism  $P(f)$  of  $D$ -instance presentations  $P(f) : P(c') \rightarrow P(c)$  for every function symbol  $f : c \rightarrow c'$  in  $C$ , such that if  $p =_C q$ , then  $P(p) \approx P(q)$ .

Think of this as a presentation of a profunctor as a functor  $\mathcal{P} : \mathcal{C} \rightarrow \mathbf{Set}^{\mathcal{D}}$ .

# Profunctor Presentations

Given category presentations  $C, D, E$  and curried profunctor presentations  $P : C \rightarrow D$  and  $Q : D \rightarrow E$ , let  $(P \circledast Q)(c)_\Sigma$  denote the  $E$ -instance signature whose generators consist of pairs of generators  $(p : d)$  of  $P(c)$  and  $(q : e)$  of  $Q(d)$ , which we write as  $(p \otimes q : e)$ .

We extend the formal  $\otimes$  operator as follows.

If  $h$  is a path in  $E$ , then set  $(p \otimes q.h) = (p \otimes q).h$ . If  $g$  is a path in  $D$ , then set  $(p.g \otimes q) = (p \otimes Q(g)(q))$ . Let  $(P \circledast Q)(c)_E$  denote the set of equations consisting of the form

- $(s \otimes q = s' \otimes q)$  for  $s =_{P(c)} s'$  and  $q$  a generator of  $Q(d)$ ,
- $(p \otimes t = p \otimes t')$  for  $t =_{Q(d)} t'$  and  $p$  a generator of  $P(c)$ .

If  $f : c \rightarrow c'$  is a function symbol in  $C$ , let

$(P \circledast Q)(f) : (P \circledast Q)(c') \rightarrow (P \circledast Q)(c)$  be defined by

$(P \circledast Q)(f)(p \otimes q) = (P(f)(p) \otimes q)$ .

**Thm**[MMR]: Given curried profunctor presentations  $P : C \rightarrow D$  and  $Q : D \rightarrow E$ , there is an isomorphism

$$\mu : \langle P \rangle \odot \langle Q \rangle \rightarrow \langle P * Q \rangle$$

of profunctors from  $\langle C \rangle$  to  $\langle E \rangle$ .

Furthermore, if  $P$  and  $Q$  are finite curried prof. pres. then clearly  $\langle P * Q \rangle$  is a finite curried prof. pres.

**Corollary:** The class of finitely curried presentable profunctors is closed under composition.



# Profunctor Presentations

So we know that curried profunctor presentations compose. However, they are not as nice to work with as uncurried profunctor presentations. It would be convenient to know what class of uncurried profunctor presentations can be turned into curried ones.

If  $P : C \multimap D$  is a curried profunctor presentation, let  $\overline{P}$  denote the uncurried profunctor presentation with function symbols  $\overline{p} : c \rightarrow d$  for every generator  $(p : d)$  of  $P(c)$ . Equations are defined similarly.

We obtain a functor  $\overline{(-)} : \mathbf{Curr}(C, D) \rightarrow \mathbf{UnCurr}(C, D)$ .

# Profunctor Presentations

Let  $P : C \leftrightarrow D$  be an uncurried profunctor presentation. By a **short left path** we mean a path in  $P$  of the form  $f.p$  where  $f$  is a  $C$ -function symbol and  $p$  is a profunctor function symbol. By a **right path** we mean one of the form  $p.g$ , where  $g$  is a path in  $D$ .

We say that  $P$  is **nongenerative** if for every short left path  $\ell$  in  $P$  there exists a right path  $r$  such that  $\ell \approx_P r$ .

Let  $P^c$  denote the  $D$ -instance with generators  $(p : d)$  for every profunctor function symbol  $p : c \rightarrow d$ , and equations

$$P_E^c = \{(r = r') \in P_E : s(r) = s(r') = c\}.$$

We say that  $P$  is **conservative** if for every pair  $t, t' : c \rightarrow d$  of right cross-paths in  $P$ , if  $t \approx_P t'$ , then  $t \approx_{P^c} t'$ . Basically we can prove they are equal using only symbols on the right side of  $P$ .

# Profunctor Presentations

For every sort  $c$  in  $C$ , there is an inclusion functor  $\iota^c : \langle P^c \rangle \rightarrow \langle P \rangle$ .

**Prop**[MMR]:  $P$  is conservative iff  $\iota^c$  is faithful for every  $c$ , and it is nongenerative iff  $\iota^c(c, d) : \langle P^c \rangle(c, d) \rightarrow \langle P \rangle(c, d)$  is surjective for every  $d \in D$ .

Call an uncurried profunctor presentation  $P$  **curryable** if it is conservative and nongenerative. Let **Cble**( $C, D$ ) denote the subcategory of **UnCurr**( $C, D$ ) on the curryable profunctor presentations with rightward morphisms<sup>1</sup>.

**Thm**[MMR]: The functor  $\overline{(-)} : \mathbf{Curr}(C, D) \rightarrow \mathbf{UnCurr}(C, D)$  lands in **Crble**( $C, D$ ). If we restrict the codomain to **Crble**( $C, D$ ), then  $\overline{(-)}$  becomes an equivalence of categories.

---

<sup>1</sup>Maps  $F : P \rightarrow P'$  of uncurried prof. pres. that send cross-paths to right cross-paths.

Thus we have characterized precisely those uncurried profunctor presentations that are equivalent to the curried ones!

We believe that this is a great starting point for investigating more deeply presentations of categorical structures.

Thank you so much for your patience and attention!

## Bonus Fact:

The category presentations, maps of category presentations and curried profunctor presentations form a double category  $\mathbf{Curr}$ , quotienting this in an appropriate way gives a double category  $\mathbf{Curr}_{\approx}$  that is equivalent to the double category  $\mathbf{Prof}$ .

## References

---

- [MMR24] Joshua Meyers, Emilio Minichiello, and Gabriel Goren Roig. **Presenting Profunctors**. 2024. arXiv: [2404.01406](https://arxiv.org/abs/2404.01406) [math.CT].