Profunctors, categorification, and monoidal categories

Kamil Oratowski

Fizyka Techniczna

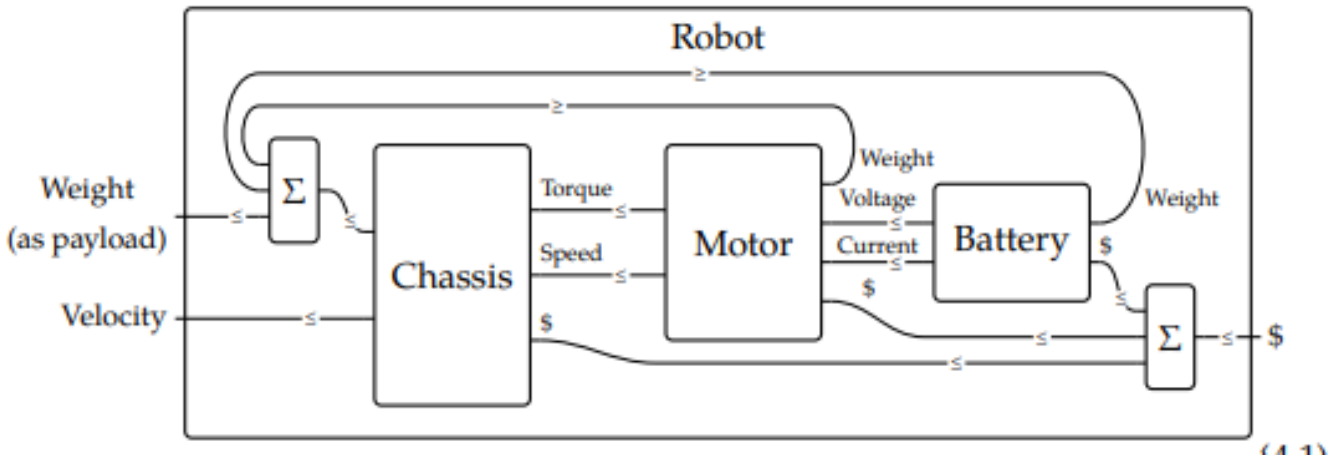
11 lutego 2020

Can we build it?

Projektujac system na duża skale, łaczy sie wiele różnych dziedzin wiedzy specjalistycznej aby pracować nad jednym projektem. W ten sposób cały zespół projektowy jest podzielony na wiele podgrupy, z których każda pracuje nad podprojektem. Podprojekt jest ponownie dzielony na podprojekty, każdy z własnym zespołem. Jeden może odnosić sie do rodzaju zhierarchizowanego procesu projektowania jako wspólnego projektowania lub współprojektowania. Rozważ tylko jeden poziom tej hierarchii: projekt i zestaw zespołów pracujacych . Każdy zespół powinien zapewniać zasoby - czasem nazywane „funkcjonalnościami". Inne zespoły projektowe musza mieć możliwość planowania i pracy niezależnie od siebie w celu postepów, który należy poczynić. Jednak decyzje projektowe podjete przez jedna grupe wpływaja na projekt i decyzje, które moga podjać inni. Połaczenie zależności i niezależności ma kluczowe znaczenie dla postepu wykonania, a jednak może powodować poważne problemy. Gdy zespół wymaga wiecej zasobów niż pierwotnie oczekiwał, że bedzie wymagał.

Wpływaja one na sasiednie drużyny: jeśli drużyna A wymaga teraz wiecej niż pierwotnie twierdziła, to zespół B może być zmuszony do zmiany projektu, co z kolei może wpłynać na zespół C. Te powiadomienia o zmianie projektu moga przenikać przez system za pośrednictwem petli sprzeżenia zwrotnego gdy zespół wymaga wiecej zasobów niż pierwotnie oczekiwał, że bedzie wymagał, lub jeśli nie może zapewnić zasobów, które pierwotnie wymagał. Jako przykład rozważ problem projektowy polegajacy na stworzeniu robota do przenoszenia pewnego obciażenia z pewna predkościa. Planista najwyższego poziomu dzieli problem na trzy zespoły projektowe: zespół podwozia, zespół silnikowy i zespół akumulatorowy. Każda z tych drużyn mogłaby sie rozpaść ale wiele cześci i procesów sie powtarza, ale pozostańmy na najwyższym poziomie i zastanówmy sie wyprodukowane zasoby i zasoby wymagane przez każdy z naszych trzech zespołów. Podwozie w pewnym sensie zapewnia cała funkcjonalność - przenosi ładunek na predkość - ale wymaga to pewnych rzeczy. Do zbudowania potrzeba pieniedzy, oczywiście, ale bardziej do rzeczy, wymaga źródła momentu obrotowego i predkości. To sa zasilany przez silnik, który z kolei potrzebuje napiecia i pradu z akumulatora. Silnik i akumulator kosztuja pieniadze.. Powstaje petla sprzeżenia zwrotnego: podwozie musi przenosić cały cieżar, nawet cześci zasilajacych podwozie. Cieższy bateria może dostarczyć wiecej energii do zasilania podwozia, ale czy jest warta dodatkowej mocy i cieższego ładunku.

Na poniższym zdjeciu każda cześć - podwozie, silnik, akumulator i robot - jest pokazana jako pudełko z portami po lewej i prawej stronie. Funkcje lub zasoby wytwarzane przez cześć jest pokazana jako porty po lewej stronie pudełka, a zasoby wymagane przez cześć jest pokazana jako porty po prawej stronie.



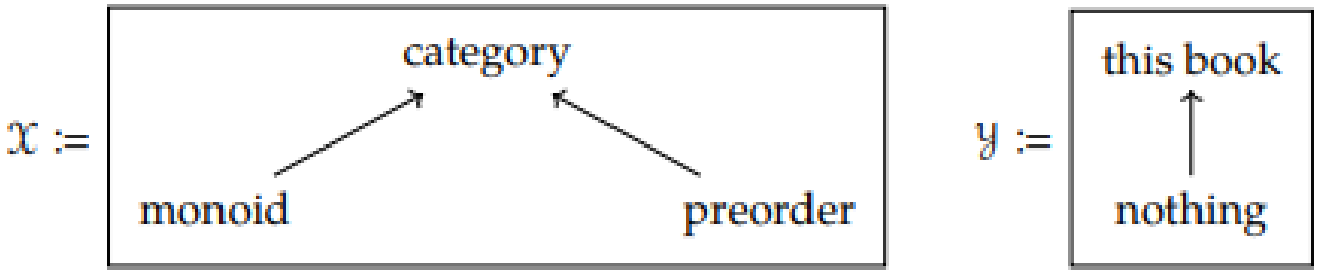
Pola oznaczone Σ odpowiadaja wejściowym wartościom sumujacym. Te pola nie powinny być zaprojektowane, ale zobaczymy później, że łatwo pasuja do tych samych ram koncepcyjnych. Podwozie wymaga momentu obrotowego, a silnik musi wytwarzać co najmniej taki moment obrotowy. Aby sformalizować to nieco bardziej, wywołajmy diagramy takie jak powyższe diagramy wspólnego projektowania. Każdy z drutów na schemacie współprojektowania reprezentuje preorder zasobów.. Ogólnie rzecz biorac, te zamówienia wstepne nie musza być zamówienia liniowe, chociaż w powyższych przypadkach każdy bedzie prawdopodobnie odpowiadał rzedowi liniowemu: 10 USD 20 USD, 5 W 6 W i tak dalej.

Relacje wykonalności definiuja zatem funkcje Φ : P R Bool. Dla funkcji Φ: P R Bool ma sens jako relacja wykonalności, jednak sa dwa warunki: (a) Jeśli Φ (p, r) prawda i p 0 p, nastepnie Φ (p 0 , r) prawda. (b) Jeśli Φ (p, r) prawda i r r 0 nastepnie Φ (p, r 0) prawdziwe. Te warunki, które zobaczymy ponownie w definicji 4.2, mówia, że jeśli potrafisz produkować p danych zasobów r, możesz (a) również produkować mniej p 0 p przy tych samych zasobach r oraz (b) produkuja również p, biorac pod uwage wiecej zasobów r 0 r. Zobaczymy, że te dwa warunki sa sformalizowane przez wymaganie, aby była monotoniczna mapa P op R Bool

Feasibility relationships as Bool-profunctors

Teoria wspólnego projektowania opiera sie na zamówieniach przedpremierowych: każdy zasób - np. predkość, moment obrotowy lub $ - ma strukture przedsprzedaży. Rzad x y reprezentuje dostepność podanego x y, to znaczy, że ilekroć masz y, masz także x. Na przykład w naszej przedsprzedaży moc, jeśli 5 W 10 W, oznacza to, że ilekroć otrzymamy 10 W, domyślnie również mieć 5 W. Powyżej nazywaliśmy to porzadkiem od mniej przydatnego do bardziej użytecznego: jeśli x jest zawsze dostepne dla y, wtedy x jest mniej użyteczne niż y.

Wiemy, że przedsprzedaż X można uznać za kategorie Bool. Biorac pod uwage x, y X, mamy X (x, y) B; ta wartość odpowiada twierdzeniu „x jest dostepny biorac pod uwage y ”, oznaczajac to jako prawde lub fałsz. Naszym celem jest postrzeganie relacji wykonalności jako Bool profunctors, które sa szczególnym przypadkiem czegoś, co nazywa sie wzbogaconymi profunctors.



1. Narysuj diagram Hasse'a dla przedsprzedaży Xop Y.
2. Zapisz profunctor : X Y i, czytajac (x, y) true, „moja ciocia może wyjaśnić x, biorac pod uwage y ", podaj interpretacje faktu, że pierwowzór prawdy tworzy górny zbiór w Xop Y. Aby uogólnić pojecie relacji wykonalności, musimy zauważyć, że symetryczny preoid monoidalny Bool ma wiecej struktury niż tylko symetryczny monoidalny To znaczy, że ma wszystko łaczy i operacje zamkniecia, która napiszemy : B B B. Z definicji to operacja spełnia właściwość, która ma dla wszystkich b, c, d B b c d iff b (c d).

The operation $\Rightarrow$ is given by the following table:

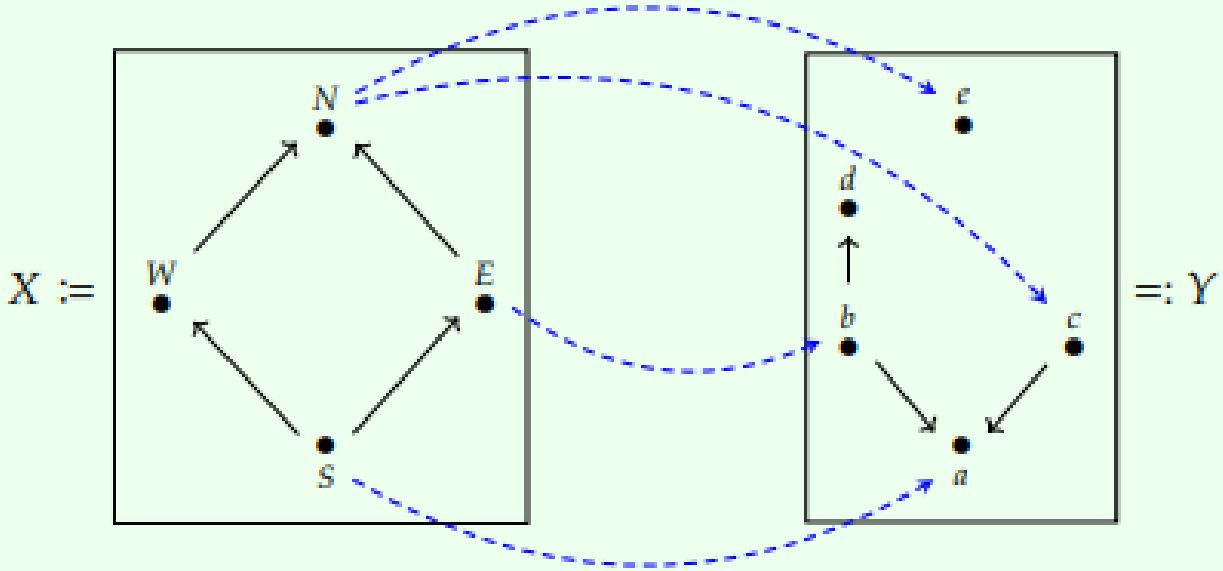
c	d	$c \Rightarrow d$
`true`	`true`	`true`
`true`	`false`	`false`
`false`	`true`	`true`
`false`	`false`	`true`

V-profunctors

Definition 4.8. Let $\mathcal{V} = (V, \leq, I, \otimes)$ be a (unital commutative) quantale,[1] and let $\mathcal{X}$ and $\mathcal{Y}$ be $\mathcal{V}$-categories. A $\mathcal{V}$-*profunctor* from $\mathcal{X}$ to $\mathcal{Y}$, denoted $\Phi\colon \mathcal{X} \nrightarrow \mathcal{Y}$, is a $\mathcal{V}$-functor

$$\Phi\colon \mathcal{X}^{\mathrm{op}} \times \mathcal{Y} \to \mathcal{V}.$$

Wiemy, że definicja jest dość abstrakcyjna. . Porozmawiajmy Definicja w sprawie V Bool. Jednym ze sposobów na wyobrażenie sobie Bool-profunctora Φ: X Y jest już dostepny warunki budowy mostów miedzy dwoma miastami. Przypomnij sobie, że zamówienie w przedsprzedaży (kategoria Bool) można narysować za pomoca diagramu Hassego. Bedziemy myśleć o zamówieniu przedpremierowym jako o mieście i każdym z nich wierzchołek w nim jako jakiś punkt zainteresowania. Strzałka A B na schemacie Hassego oznacza że istnieje sposób na przejście z punktu A do punktu B w mieście. Czym wiec jest profunctor? Profunctor to tylko kilka mostów łaczacych punkty w jednym mieście z punktami w inne. Zobaczmy konkretny przykład. Oto zdjecie Bool-profunctora Φ: X Y:



Zarówno X, jak i Y sa zamówieniami przedpremierowymi, np. z W N ib a. Z mostów pochodzacych profonctorami w kolorze niebieskim, można teraz korzystać z obu ścieżek w obrebie miast i mostów do dostać sie z punktów w mieście X do punktów w mieście Y. Na przykład, ponieważ istnieje ścieżka z N. do e i E do a, mamy Φ (N, e) true i (E, a) true. Z drugiej strony, ponieważ nie ma ścieżki od W do d, mamy Φ (W, d) fałsz.

Exercise 4.12. We can express Φ as a matrix where the (m, n)th entry is the value of $\Phi(m, n) \in \mathbb{B}$. Fill out the **Bool**-matrix:

Φ	a	b	c	d	e
N	?	?	?	?	`true`
E	`true`	?	?	?	?
W	?	?	?	`false`	?
S	?	?	?	?	?

$\Diamond$

Fill out the **Cost**-matrix:

Φ	x	y	z
A	?	?	20
B	11	?	?
C	?	17	?
D	?	?	?

Remark 4.16 (Computing profunctors via matrix multiplication). We can give an algorithm for computing the above distance matrix using matrix multiplication. First, just like in Eq. (2.59), we can begin with the labelled graphs in Eq. (4.14) and read off the matrices of arrow labels for X, Y, and Φ:

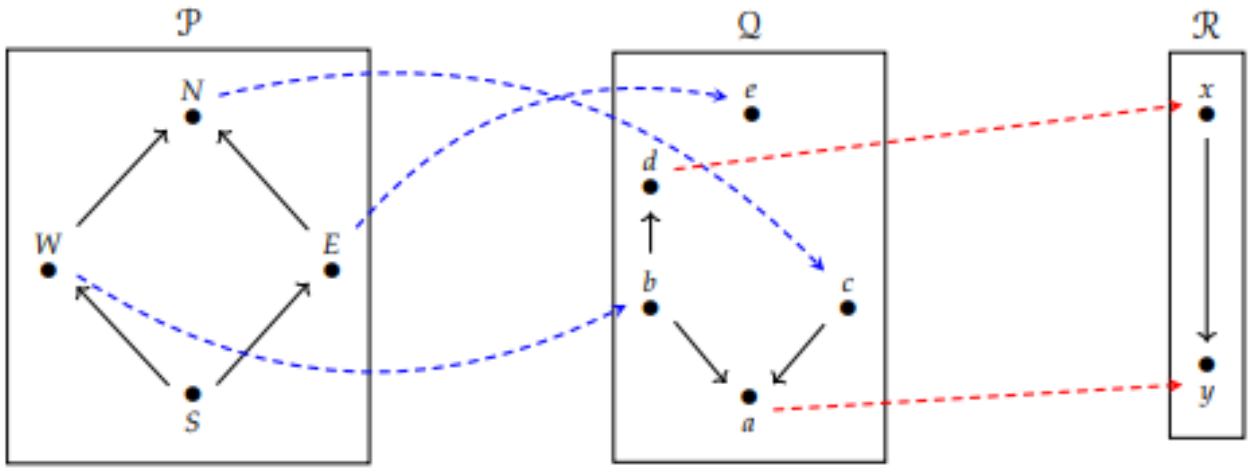
M_X	A	B	C	D
A	0	∞	3	∞
B	2	0	∞	5
C	∞	3	0	∞
D	∞	∞	4	0

M_Φ	x	y	z
A	∞	∞	∞
B	11	∞	∞
C	∞	∞	∞
D	∞	9	∞

M_Y	x	y	z
x	0	4	3
y	3	0	∞
z	∞	4	0

Categories of profunctors

Jeśli relacje wykonalności maja być morfizmami, musimy podać formułe do komponowania dwa z nich w serii. Wyobraź sobie, że masz miasta P, Q i R, a także mosty - a zatem macierze wykonalności - łaczace te miasta, powiedzmy Φ: P Q i Ψ: Q R.



(4.19)

The feasibility matrices for Φ (in blue) and Ψ (in red) are:

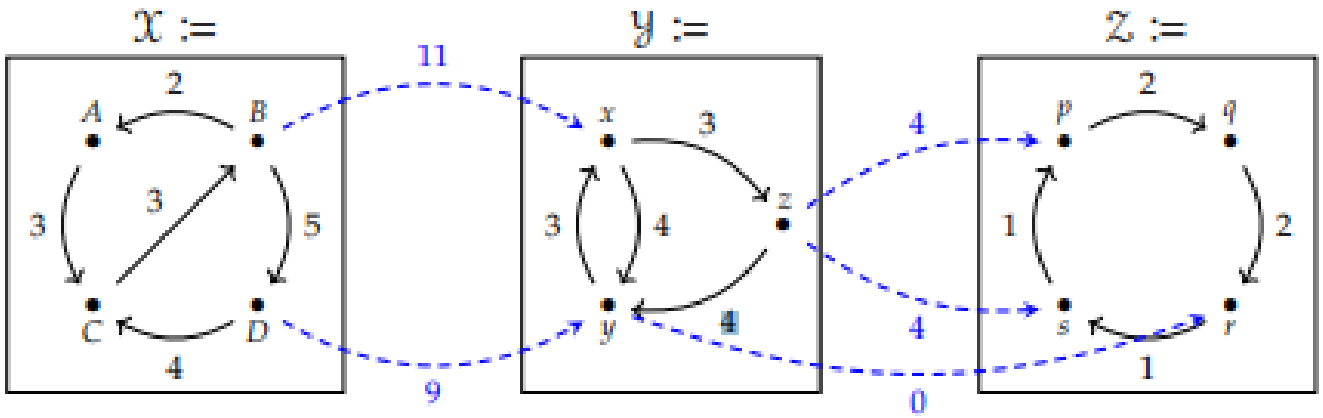
Φ	a	b	c	d	e
N	true	false	true	false	false
E	true	false	true	false	true
W	true	true	true	true	false
S	true	true	true	true	true

Ψ	x	y
a	false	true
b	true	true
c	false	true
d	true	true
e	false	false

Definition 4.21. Let $\mathcal{V}$ be a quantale, let $\mathcal{X}$, $\mathcal{Y}$, and $\mathcal{Z}$ be $\mathcal{V}$-categories, and let $\Phi\colon \mathcal{X} \nrightarrow \mathcal{Y}$ and $\Psi\colon \mathcal{Y} \nrightarrow \mathcal{Z}$ be $\mathcal{V}$-profunctors. We define their *composite*, denoted $\Phi \fatsemi \Psi\colon \mathcal{X} \nrightarrow \mathcal{Z}$ by the formula

$$(\Phi \fatsemi \Psi)(p, r) = \bigvee_{q \in Q} \big(\Phi(p, q) \otimes \Psi(q, r)\big).$$

22. Consider the **Cost**-profunctors $\Phi\colon \mathcal{X} \nrightarrow \mathcal{Y}$ and $\Psi\colon \mathcal{Y} \nrightarrow \mathcal{Z}$ shown below:



Fill in the matrix for the composite profunctor:

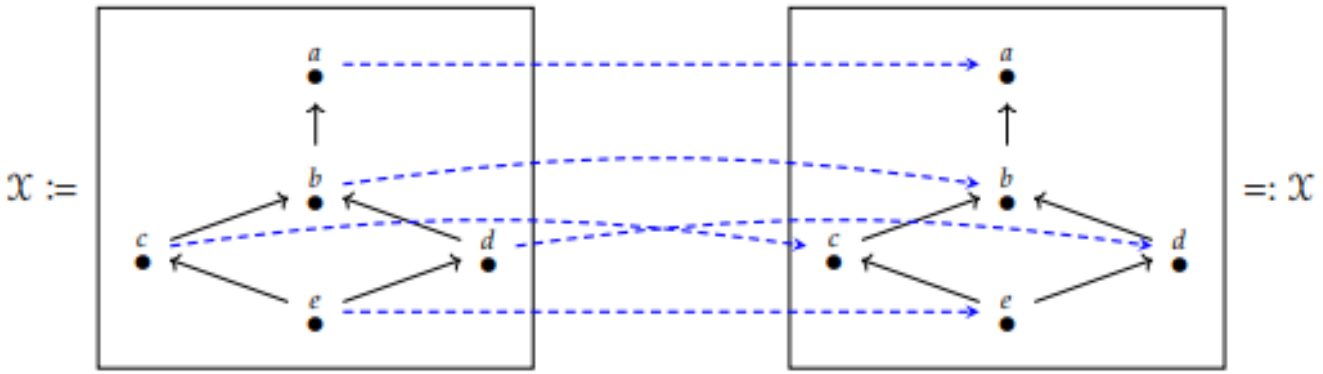
$\Phi \fatsemi \Psi$	p	q	r	s
A	?	24	?	?
B	?	?	?	?
C	?	?	?	?
D	?	?	9	?

The categories V-Prof and Fea

Reguła składu sugeruje kategorie, a tak naprawde istnieje kategoria, w której obiekty sa kategoriami Bool, a morfizmy sa profesorami Bool. Aby to zrobić pracuja bardziej ogólnie, jednak musimy dodać jeden warunek techniczny. . Preorder szkieletowy jest również znany jako poset. Mówimy kwantowy jest szkieletowy, jeśli jego podstawa jest szkielet; Bool i Cost to kwanty szkieletowe

How do we interpret this? Recall that, by Definition 2.46, $\mathcal{X}$ already assigns to each pair of elements $x, y \in \mathcal{X}$ an hom-object $\mathcal{X}(x, y) \in \mathcal{V}$. The unit profunctor $\mathrm{U}_{\mathcal{X}}$ just assigns each pair (x, y) that same object.

In the **Bool** case the unit profunctor on some preorder $\mathcal{X}$ can be drawn like this:



Lemma 4.27. Composing any profunctor $\Phi\colon \mathcal{P} \to \mathcal{Q}$ with either unit profunctor, $\mathrm{U}_{\mathcal{P}}$ or $\mathrm{U}_{\mathcal{Q}}$, returns Φ:

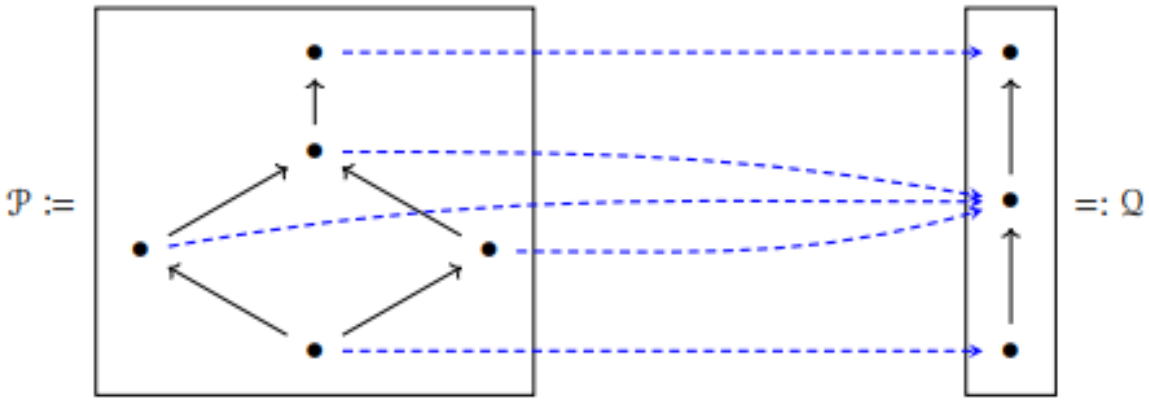
$$\mathrm{U}_{\mathcal{P}} \fatsemi \Phi = \Phi = \Phi \fatsemi \mathrm{U}_{\mathcal{Q}}$$

Fun profunctor facts: companions, conjoints, collages

Definition 4.34. Let $F\colon \mathcal{P} \to \mathcal{Q}$ be a $\mathcal{V}$-functor. The *companion of* F, denoted $\widehat{F}\colon \mathcal{P} \nrightarrow \mathcal{Q}$ and the *conjoint of* F, denoted $\check{F}\colon \mathcal{Q} \nrightarrow \mathcal{P}$ are defined to be the following $\mathcal{V}$-profunctors:

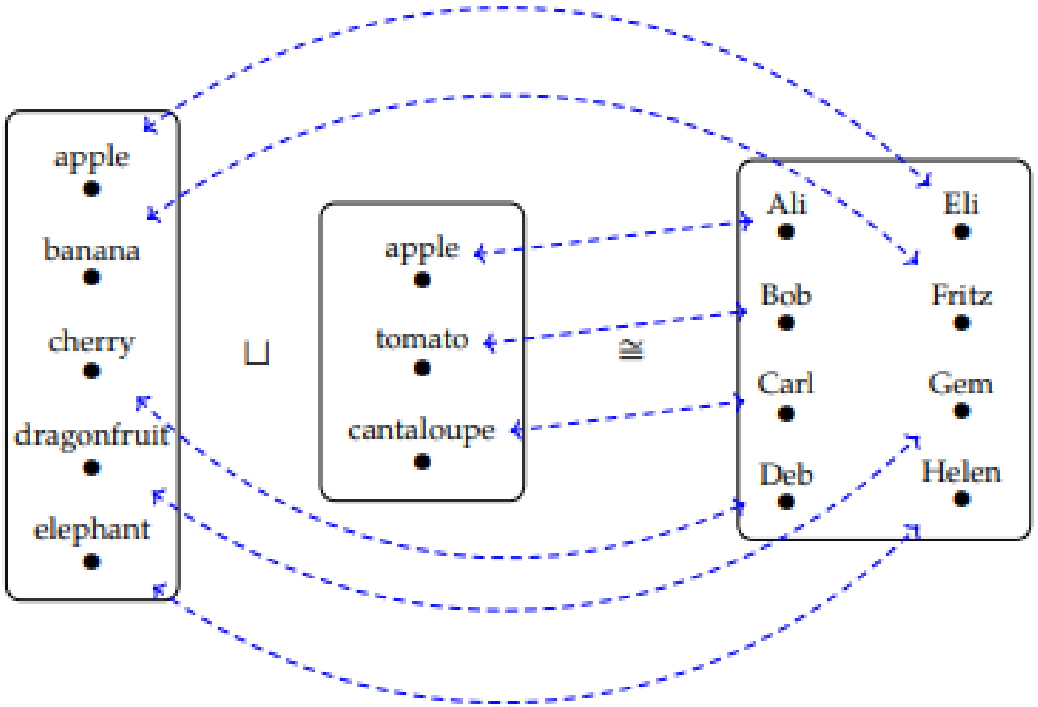
$$\widehat{F}(p,q) \coloneqq \mathcal{Q}(F(p),q) \quad \text{and} \quad \check{F}(q,p) \coloneqq \mathcal{Q}(q,F(p))$$

Let's consider the **Bool** case again. One can think of a monotone map $F\colon \mathcal{P} \to \mathcal{Q}$ as a bunch of arrows, one coming out of each vertex $p \in P$ and landing at some vertex $F(p) \in Q$.



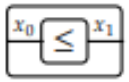
Categorification

Ogólna idea kategoryzacji polega na tym, że bierzemy coś, co wiemy, i dodajemy strukture do tego, aby to, co poprzednio było własnościa, stało sie struktura. Robimy to w taki sposób że możemy odzyskać rzecz, która sklasyfikowaliśmy, zapominajac o tej w nowej strukturze. Przykład Podstawowa arytmetyka dotyczy właściwości liczb naturalnych N, takich jak fakt że 5 + 3 8. Jednym ze sposobów kategoryzacji N jest użycie kategorii FinSet zbiorów skończonych i funkcje. Aby uzyskać kategoryzacje, zastepujemy 5, 3 i 8 na zestawy tych wielu elementów, powiedzmy 5= (jabłko, banan, wiśnia, owoc smoka, słoń), 3= jabłko, pomidor, kantalupa i 8 =(Ali, Bob, Carl, Deb, Eli, Fritz, Gem, Helen) odpowiednio. Zastepujemy również + rozłacznym połaczeniem zbiorów t i brutalna właściwościa równość ze struktura izomorfizmu.


apple
banana
cherry
dragonfruit
elephant
⊔
apple
tomato
cantaloupe
≅
Ali
Bob
Carl
Deb
Eli
Fritz
Gem
Helen

A reflection on wiring diagrams

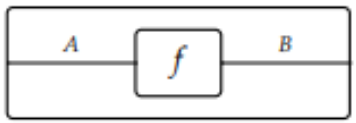
Suppose we have a preorder. We introduced a very simple sort of wiring diagram in Section 2.2.2. These allowed us to draw a box



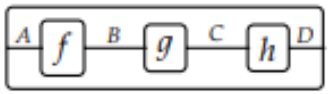
whenever $x_0 \leq x_1$. Chaining these together, we could prove facts in our preorder. For example

provides a proof that $x_0 \leq x_3$ (the exterior box) using three facts (the interior boxes), $x_0 \leq x_1$, $x_1 \leq x_2$, and $x_2 \leq x_3$.

As categorified preorders, categories have basically the same sort of wiring diagram as preorders—namely sequences of boxes inside a box. But since we have replaced the fact that $x_0 \leq x_1$ with the structure of a *set* of morphisms, we need to be able to label our boxes with morphism names:

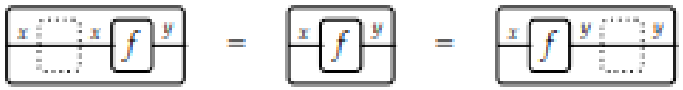


Suppose given additional morphisms $g\colon B \to C$, and $h\colon C \to D$. Representing these each as boxes like we did for f, we might be tempted to stick them together to form a new box:



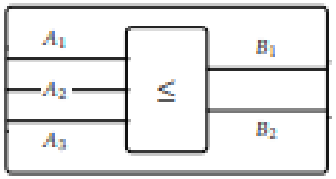
Similarly, the identity morphism on an object x is drawn as on the left below, but we will see that it is not harmful to draw id_x in any of the following three ways:

By Definition 3.6 the morphisms in a category satisfy two properties, called the unitality property and the associativity property. The unitality says that $\mathrm{id}_x \fatsemi f = f = f \fatsemi \mathrm{id}_y$ for any $f : x \to y$. In terms of diagrams this would say

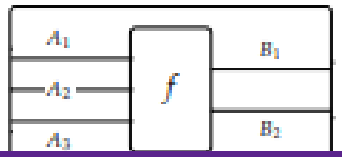


This means you can insert or discard any identity morphism you see in a wiring diagram. From this perspective, the coherence laws of a category—that is, the associativity law and the unitality law—are precisely what are needed to ensure we can lengthen and shorten wires without ambiguity.

In Section 2.2.2, we also saw wiring diagrams for monoidal preorders. Here we were allowed to draw boxes which can have multiple typed inputs and outputs, but with no choice of label (always $\leq$):

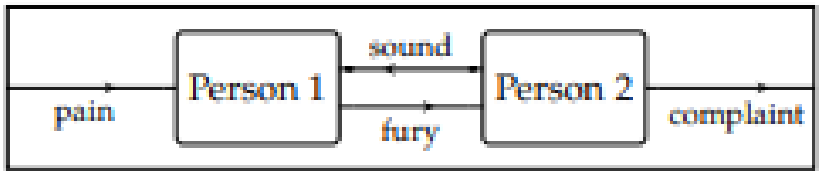


If we combine these ideas, we will obtain a categorification of symmetric monoidal preorders: symmetric monoidal categories. A symmetric monoidal category is an algebraic structure in which we have labelled boxes with multiple typed inputs and outputs:



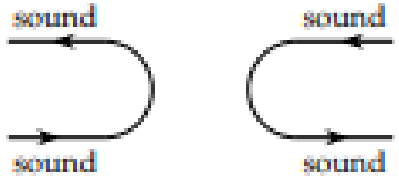
Profunctors form a compact closed category

Kompaktowe zamkniete kategorie sa symetrycznymi kategoriami monoidalnymi, z nieco wieksza struktura, która pozwala nam formalnie interpretować rodzaje informacji zwrotnych, które wystepuja podczas wspólnego projektowania problemów. Ta sama struktura pojawia sie w wielu innych dziedzinach, w tym w kwantowej mechanice i układach dynamicznych. Do kompaktowo zamknietych kategorii, nasze dodatkowe ikony pozwalaja nam zginać dane wyjściowe na dane wejściowe i odwrotnie. Aby to jednak śledzić, narysujemy strzały na naszym drucie, które moga wskazywać do przodu lub do tyłu.



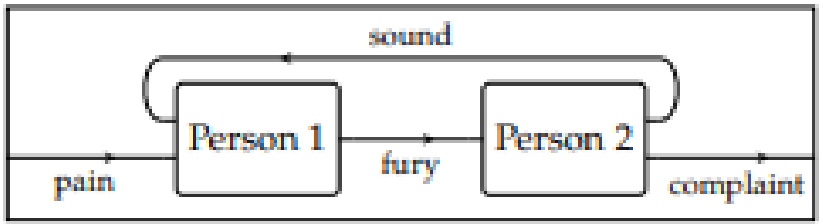
(4.56)

We then add icons—called a cap and a cup—allowing any wire to reverse direction from forwards to backwards and from backwards to forwards.



(4.57)

Thus we can draw the following



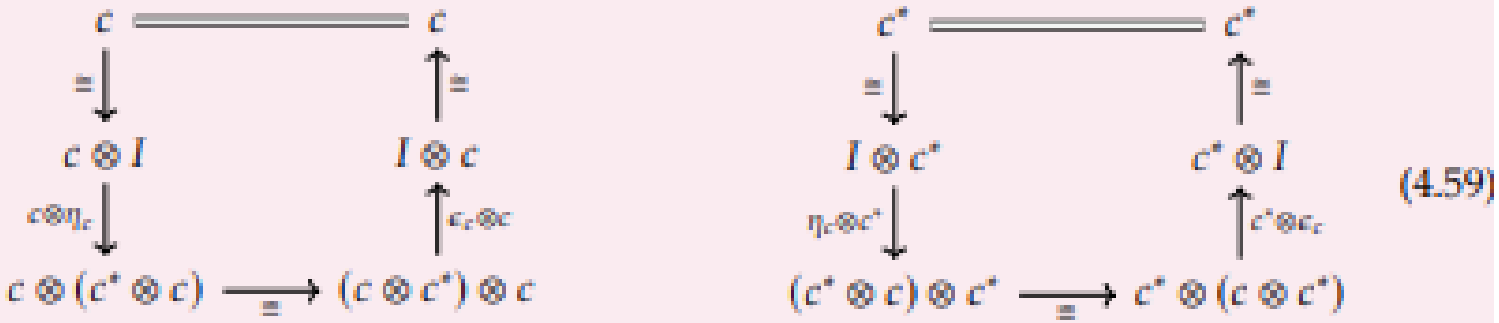
Definition 4.58. Let $(\mathcal{C}, I, \otimes)$ be a symmetric monoidal category, and $c \in \mathrm{Ob}(\mathcal{C})$ an object. A *dual for* c consists of three constituents

(i) an object $c^* \in \mathrm{Ob}(\mathcal{C})$, called the *dual of* c,

(ii) a morphism $\eta_c : I \longrightarrow c^* \otimes c$, called the *unit for* c,

(iii) a morphism $\epsilon_c : c \otimes c^* \longrightarrow I$, called the *counit for* c.

These are required to satisfy two equations for every $c \in \mathrm{Ob}(\mathcal{C})$, which we draw as commutative diagrams:



(4.59)

These equations are sometimes called the *snake equations*.

If for every object $c \in \mathrm{Ob}(\mathcal{C})$ there exists a dual c^* for c, then we say that $(\mathcal{C}, I, \otimes)$ is *compact closed*.