Linear Logic and Co-Design Notebook

Marius Furter

May 7, 2021

Contents

1 Categories in Linear Logic

1.1 The Category of Derivations

Given a sequent calculus S, we form the category of derivations in S, denoted $\mathbf{Der}_S$ as follows. The objects are well-formed sequents

$$\Gamma, A \vdash B \qquad \Gamma \vdash A \multimap B \qquad \dots$$

and the morphisms are valid derivations based on the rules of S, for example,

$$\frac{\Gamma, A \vdash B}{\Gamma \vdash A \multimap B}\ (\multimap \text{ Right})$$

Composition is given by the composition of derivations.

$$\frac{\dfrac{\Gamma \vdash B}{\Gamma, A \vdash B}\ (\text{weakening Left})}{\Gamma \vdash A \multimap B}\ (\multimap \text{ Right}) \quad \rightsquigarrow \quad \frac{\dfrac{\Gamma \vdash B}{\Gamma, A \vdash B}\ (\text{weakening Left})}{\Gamma \vdash A \multimap B}\ (\multimap \text{ Right})$$

It is clearly associative and has the "do nothing" sequent

$$\frac{\Gamma \vdash \Delta}{\Gamma \vdash \Delta}\ (\text{identity})$$

as identity.

This all works fine for unary rules. To accommodate rules of higher arity, we need to allow multiple sources for our morphisms. We could then think of $\mathbf{Der}_S$ as a multicategory (colored operad) or, alternatively, replace objects by lists of objects and introduce a monoidal product $\boxplus$ that allows us to concatenate these lists. The unit for this monoidal product is the empty list $[\,]$. We shall write lists of objects $[A, B, C]$ as $A \boxplus B \boxplus C$, that is we omit the brackets for lists of length one. For now, due to a current lack a familiarity with multicategories, we shall regard $\mathbf{Der}_S$ as a monoidal category. In this setting, rules of higher arity become morphisms from lists of length greater 1 to lists of length 1. For example, the cut rule

$$\frac{\Gamma, A \vdash B \qquad \Delta, B \vdash C}{\Gamma, \Delta, A \vdash C}\,(\text{Cut})$$

is a morphism $(\Gamma, A \vdash B) \boxplus (\Delta, B \vdash C) \to \Gamma, \Delta, A \vdash C$. The axiom

$$\frac{}{A \vdash A}\,(\text{Axiom})$$

is a morphism $[\,] \to A \vdash A$.

We may now view the category $\mathbf{Der}_S$ as being generated by the rules of the sequent calculus S: We fix some set of propositions $\Gamma, \Delta, \Lambda, \ldots$ and consider all sequents that can be formed from these. We then form all finite lists of such sequents and add morphisms according to the rules of S. Finally $\mathbf{Der}_S$ is obtained by closing this structure under composition.

Question 1.1. *Read about operads / multicategories in Leinster's book Chapter 2. Think about the algebras of* $\mathbf{Der}_S$.

1.2 The Category of Propositions

The category of proposition $\mathbf{Prop}_S$ has a objects propositions $\Gamma, \Delta, \Lambda, \ldots$ and as morphisms proofs of sequents $\Gamma \vdash \Delta$. The compostion of morphisms is given by the cut rule

$$\frac{A \vdash B \qquad B \vdash C}{A \vdash C}\,(\text{Cut})$$

which turn two proofs into one. We note that cut rule depends on the sequent calculus. For example, in intuitionistic linear logic it is

$$\frac{\Gamma \vdash A \qquad \gamma_1, A, \gamma_2 \vdash B}{\gamma_1, \Gamma, \gamma_2 \vdash B}\,(\text{Cut})$$

Our basic rule is obtained with empty contexts γ_1, γ_2.

1.3 The Relation between Der and Prop

To start with suppose that our calculus S consists only of unary rules. In this case $\mathbf{Der}_S$ is simply a category with a special object for the empty sequent $[\,]$. We observe that the arrows of $\mathbf{Prop}_S$ are special derivations: An arrow $\Gamma \to \Delta$ in $\mathbf{Prop}_S$ corresponds to an arrow in $\mathbf{Der}_S$ of the form $[\,] \to \Gamma \vdash \Delta$. Therefore $\mathbf{Prop}_S(\Gamma, \Delta) \cong \mathbf{Der}_S([\,], \Gamma \vdash \Delta)$. Moreover, each element of $\mathbf{Der}_S(\Gamma \vdash \Delta, A \vdash B)$ gives a function $\mathbf{Prop}_S(\Gamma, \Delta) \to \mathbf{Prop}_S(A, B)$, obtained by composing derivations. This yields a functor $F : \mathbf{Der}_S \to \mathbf{Set}$ mapping objects $(\Gamma \vdash \Delta) \mapsto \mathbf{Prop}_S(\Gamma, \Delta)$ and derivations $\Gamma, \Delta \to A, B$ to the induced function $\mathbf{Prop}_S(\Gamma, \Delta) \to \mathbf{Prop}_S(A, B)$. Indeed, functoriality is immediate.

We extend the construction above to the case where S has rules of arbitrary arity. In this case the objects in $\mathbf{Der}_S$ are lists of sequents. Because $\boxplus^n[\,] \cong [\,]$ we still have the correspondence $\mathbf{Prop}_S(\Gamma, \Delta) \cong \mathbf{Der}_S([\,], \Gamma \vdash \Delta)$. Next we note that if we have an arrow $d : A_1 \boxplus \ldots \boxplus A_n \to B_1 \boxplus \ldots \boxplus B_m$ then $n \geq m$ because the rules of S are many to one. Moreover, we can regard d as consisting of m parallel derivations $d_i : A_{i_1} \boxplus \ldots \boxplus A_{i_r} \to B_i$. Writing $A_{i_j} = \Gamma_{i_j} \vdash \Delta_{i_j}$ and $B_i = \Psi_i \vdash \Omega_i$, we see that each d_i induces a function

$$\mathbf{Prop}_S(\Gamma_{i_1}, \Delta_{i_1}) \times \ldots \times \mathbf{Prop}_S(\Gamma_{i_r}, \Delta_{i_r}) \to \mathbf{Prop}_S(\Psi_i, \Omega_i).$$

Hence d induces a function

$$\prod_i \mathbf{Prop}_S(\Gamma_{i_1}, \Delta_{i_1}) \times \ldots \times \mathbf{Prop}_S(\Gamma_{i_r}, \Delta_{i_r}) \to \prod_i \mathbf{Prop}_S(\Psi_i, \Omega_i).$$

We now face the problem that the decomposition of the A_i is not the same for every derivation with the same source and target as d. However, writing $A_j = \Gamma_j \vdash \Delta_j$, we have a natural iso

$$\prod_i \mathbf{Prop}_S(\Gamma_{i_1}, \Delta_{i_1}) \times \ldots \times \mathbf{Prop}_S(\Gamma_{i_r}, \Delta_{i_r}) \cong \prod_{j=1}^{n} \mathbf{Prop}_S(\Gamma_j, \Delta_j)$$

by reordering the big product. This suggests that we assign a list of sequents $(\Gamma_1 \vdash \Delta_1) \boxplus \ldots \boxplus (\Gamma_n \vdash \Delta_n)$ to the product $\prod_{j=1}^{n} \mathbf{Prop}_S(\Gamma_j, \Delta_j)$. A morphism

$$f : (\Gamma_1 \vdash \Delta_1) \boxplus \ldots \boxplus (\Gamma_n \vdash \Delta_n) \to (\Psi_1 \vdash \Omega_1) \boxplus \ldots \boxplus (\Psi_m \vdash \Omega_m)$$

is then assigned to the function

$$\tilde{f} : \prod_{j=1}^{n} \mathbf{Prop}_S(\Gamma_j, \Delta_j) \to \prod_i \mathbf{Prop}_S(\Psi_i, \Omega_i)$$

which it induces by the procedure described above, possibly composing with the reordering isomorphism.

Explicitly, we can describe $\tilde{f}$ as follows: Given a tuple of proofs of sequents

$$(\Gamma_1 \vdash \Delta_1, \ldots, \Gamma_n \vdash \Delta_n),$$

regard them as a derivation

$$[\,] \to (\Gamma_1 \vdash \Delta_1) \boxplus \ldots \boxplus (\Gamma_n \vdash \Delta_n)$$

and compose with f to get a derivation

$$[\,] \to (\Psi_1 \vdash \Omega_1) \boxplus \ldots \boxplus (\Psi_m \vdash \Omega_m)$$

which corresponds to a tuple of proofs of sequents

$$(\Psi_1 \vdash \Omega_1, \ldots, \Psi_m \vdash \Omega_m).$$

The functoriality of this assignment is again immediate. Hence, we also get a functor $F : \mathbf{Der}_S \to \mathbf{Set}$ in the case of a general sequent calculus S. Furthermore, F maps monoidal products in $\mathbf{Der}_S$ to cartesian products in $\mathbf{Set}$. We have $F([\,]) = \{*\}$ by definition, which aligns with the intuition that there is exactly one trivial proof of the empty sequent. Therefore F is a strict monoidal functor into $(\mathbf{Set}, \times, \{*\})$.

Question 1.2. *Regarding* $\mathbf{Der}_S$ *as an operad, can we describe the above as an operad algebra?*

We conclude by observing that for $A_i = \Gamma_i \vdash \Delta_i$ we have

$$\mathbf{Der}_S([\,], A_1 \boxplus \ldots \boxplus A_n) \cong \mathbf{Prop}_S(\Gamma_1, \Delta_1) \times \ldots \times \mathbf{Prop}_S(\Gamma_n, \Delta_n) = F(A_1 \boxplus \ldots \boxplus A_n)$$

so in fact F is represented by $[\,]$. Furthermore, we have defined the way F acts on arrows in terms of this isomorphism, so the isomorphism is natural. Hence $F \cong \mathrm{Hom}([\,], -)$. By Yoneda we have that for any element $x \in F(A)$ there is a unique morphism $f : [\,] \to A$ such that $F(f)(*) = x$. This is saying that proofs correspond precisely to derivations starting from $[\,]$.

2 Linear Logic and Co-Design

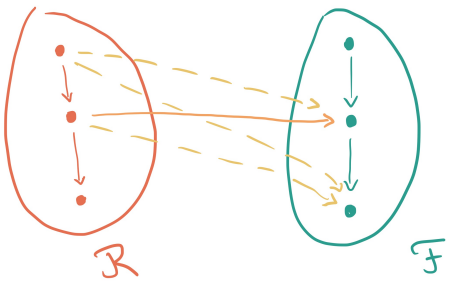
2.1 Basic Definitions

Let $\mathcal{R}$ and $\mathcal{F}$ be posets. $\mathcal{R}$ represents resources and $\mathcal{F}$ represents functionalities. For $a, b \in \mathcal{R}$, we interpret $a \leq b$ as meaning "if I have a, then I also have b". For example, $2\$ \leq 1\$$. This is the opposite convention that is currently in use, but better suits the Linear Logic interpretation. To avoid confusion, we write $a \to b$, or $a \vdash b$, instead of $a \leq b$. This also fits nicely with the fact that we will be regarding the posets as categories.

In Co-Design we consider what functionalities we can obtain from given resources.

Definition 2.1 (Feasibility Relation)**.** Let $\mathcal{R}$, $\mathcal{F}$ be posets. A *feasibility relation* $\mathcal{R} \nrightarrow \mathcal{F}$ is a monotone map $\mathcal{R}^{\mathrm{op}} \times \mathcal{F} \to \mathbf{Bool}$, where **Bool** denotes the poset generated by $\texttt{false} \to \texttt{true}$. If we regard posets as **Bool**-categories, then feasibility relations are just **Bool**-profunctors.

Remark 2.1. We can draw a feasibility relation $\mathcal{R} \nrightarrow \mathcal{F}$ as an internal diagram:



The orange and yellow arrows mean that a given pair is mapped to `true`. Absence of an arrow means the pair is mapped top `false`. Additionally, we may think of a feasibility relation as being generated by certain assignments. The solid orange arrow induces the dashed yellow arrows by composition with the arrows internal to both $\mathcal{R}$ and $\mathcal{F}$.

We can now define a category in which feasibility relations live.

Definition 2.2. The category **DP** of co-design problems has posets as objects and feasibility relations as morphisms. Composition is given by profunctor compostition. Explicitly, if we have feasibility relations $\Phi : \mathcal{X} \nrightarrow \mathcal{Y}$

and $\Psi : \mathcal{Y} \nrightarrow \mathcal{Z}$, then

$$\Phi \fatsemi \Psi(x, z) = \bigvee_{y \in \mathcal{Y}} (\Phi(x, y) \wedge \Psi(y, z)).$$

Every feasibility relation $\mathcal{R} \overset{\Phi}{\nrightarrow} \mathcal{F}$ naturally gives rise to two dual optimization problems. If we fix an $f \in \mathcal{F}$, then we can ask what the minimal set of resources $r \in \mathcal{R}$ are, such that $\Phi(r, f) = \texttt{true}$, that is (r, f) is feasible. In formulae:

$$\begin{aligned} f_{\max} &= \mathrm{Max}\{r \in \mathcal{R} \mid \Phi(r, f) = \texttt{true}\} \\ &= \{r \in \mathcal{R} \mid \Phi(r, f) = \texttt{true} \text{ and } \forall r' : \Phi(r', f) = \texttt{true} \wedge r \to r' \Rightarrow r = r'\}. \end{aligned}$$

Observe that the minimality condition implies that this set is an antichain in $\mathcal{R}$. Hence we get a function $h : \mathcal{F} \to \mathsf{A}\mathcal{R}$, $f \mapsto f_{\max}$ where $\mathsf{A}\mathcal{R}$ denotes the set of antichains of $\mathcal{R}$. We can put a partial ordering on $\mathsf{A}\mathcal{R}$ by saying $A \leq B$ iff $\downarrow A \subseteq \downarrow B$, where $\downarrow$ denotes the lower closure operator. In our convention the lower closure of a set is all of the resources / functionalities that are 'more expensive' than items in the set.

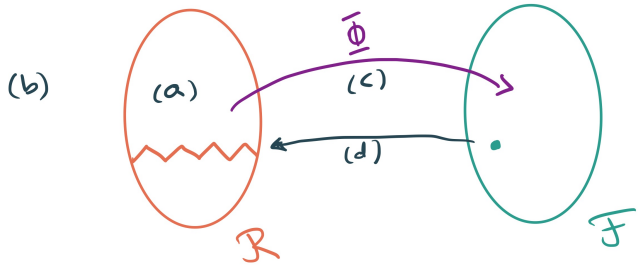
Lemma 2.1. *If all ascending chains in $\mathcal{R}$ are finite, then the map $h : \mathcal{F} \to \mathsf{A}\mathcal{R}$ is monotone. That is, if $f \to g$, then $\downarrow f_{max} \subseteq \downarrow g_{max}$.*

Proof. Let $f \to g$, and $x \in \downarrow f_{\max}$. This means $x \to r$ for some $r \in f_{\max}$. Because $\Phi(r, f) = \texttt{true}$ also $\Phi(x, g) = \texttt{true}$ using monotonicity in both arguments. Consider $\{s \in \mathcal{R} \mid x \to s \text{ and } \Phi(s, g) = \texttt{true}\}$. This set is non-empty because it contains x itself. Moreover it has to contain a maximal element, otherwise we could build an infinite ascending chain. This maximum m will be an element of $g_{\max}$, hence we have an arrow $x \to m$, which means $x \in \downarrow g_{\max}$ ⌐╚

Warning! The map h does not have to monotone in general (even if one uses upper closure or the opposite orderings). The ascending chain condition assures that one does not have arbitrarily cheap resources. To see that it is necessary consider $\mathcal{R} = (\mathbb{Z}, \leq)$, $\mathcal{F} = a \to b$ with feasibility relation $\Psi(r, f) = \texttt{true}$ iff $f = b$ or $r \leq 0$. Then $a_{\max} = \mathrm{Max}\{n \in \mathbb{Z} | \Psi(r, a) = \texttt{true}\} = 0$ and $b_{\max} = \mathrm{Max}\{n \in \mathbb{Z} | \Psi(r, b) = \texttt{true}\} = \mathrm{Max}\ \mathbb{Z} = \emptyset$. Hence $\downarrow a_{\max} = \mathbb{Z}_{\leq 0} \not\subseteq \emptyset = \downarrow b_{\max}$.

2.2 Sites to Structure

The main goal of this project is to add some of the structure of Linear Logic to the co-design framework. In principle, there are several places in the theory where one could consider implementing these concepts. Here we describe four possibilities and give some thoughts on how they may be related.



One can imagine having Linear Logic (LL) structure:

(a) internal to the resource / functionality posets. This would involve restricting the objects of **DP** to posets that can be viewed as models for LL.

(b) between objects of **DP**. This would involve turning **DP** into a model of LL.

(c) within a feasibility relation. This might involve looking at monotone maps $\mathcal{R}^{\mathrm{op}} \times \mathcal{F} \to \mathcal{V}$, where $\mathcal{V}$ is a model for LL.

(d) on the queries.

Having (b) might give (a) as well, since we can regard a poset $\mathcal{P}$ as the collage of trivial feasibility relations between singletons: If $a \to b$, then this may be seen as a feasibility relation $\Phi : \{a\} \nrightarrow \{b\}$ with $\Phi(a, b) = \texttt{true}$. If we collage these together we can reconstruct $\mathcal{P}$. For this to work we would have to describe a multi-object collage operation, but this seems possible.

The category **DP** is compact closed, which means it forms a degenerate *-autonomous category. Hence it is already a model for classical LL. However,

the 4 classical LL connectives become pairwise identified. Thus the task would be to modify **DP** in such a way, that we get distinct connectives.

Next, changing the enriching category as in (c) may also affect (b). Moreover, it will alter what type of queries are possible because there would then be more states to optimize over. On the other hand, it may be possible to complexify the types of queries that can be performed, while keeping the standard boolean feasibility relation.

2.3 Currying Queries

There is an adjunction between the exponential and the cartesian product in **Cat**:

$$\mathrm{Hom}(\mathcal{C} \times \mathcal{D}, \mathcal{E}) \cong \mathrm{Hom}(\mathcal{C}, \mathcal{E}^{\mathcal{D}}).$$

If $\mathcal{P}$ and $\mathcal{Q}$ are posets, then so is the functor category $\mathcal{Q}^{\mathcal{P}}$: If $g, h : \mathcal{P} \to \mathcal{Q}$ are functors (i.e monotone maps), there is at most one natural transformation $\alpha : g \Rightarrow h$, since there is at most one morphism $\alpha_p : g(p) \to h(p)$ for each $p \in \mathcal{P}$. Therefore, the adjunction restricts to the subcategory **Pos** of posets.

Let $\mathcal{R}, \mathcal{F}$ be posets, and let $\mathcal{R} \overset{\Phi}{\nrightarrow} \mathcal{F}$ be a feasibility relation. Denote the lower sets of $\mathcal{R}$ by $\mathsf{L}\mathcal{R}$, where $A \leq B$ iff $A \subseteq B$. We claim that $\mathbf{Bool}^{\mathcal{R}^{\mathrm{op}}} \cong \mathsf{L}\mathcal{R}$. A monotone map $h : \mathcal{R}^{\mathrm{op}} \to \mathbf{Bool}$ is the same as a monotone map $h' : \mathcal{R} \to \mathbf{Bool}^{\mathrm{op}}$. The pre-image $L_h = h'^{-1}(\mathtt{true})$ is now a lower set: If $y \in L$ and $x \to y$, then $h'(x) \to h'(y) = \mathtt{true}$, so $h'(x) = \mathtt{true}$ since $\mathtt{true}$ is the bottom in $\mathbf{Bool}^{\mathrm{op}}$. Moreover, if $g, h : \mathcal{R}^{\mathrm{op}} \to \mathbf{Bool}$ and $g \Rightarrow h$, then $g(r) \to h(r)$ in $\mathbf{Bool}^{\mathrm{op}}$ for all $r \in \mathcal{R}$. Hence, if $h(r) = \mathtt{true}$, then also $g(r) = \mathtt{true}$, which shows $L_h \subseteq L_g$, i.e. $L_h \leq L_g$ in the lower set order. Conversely, for each lower set $L \subseteq \mathcal{R}$, we can construct an monotone map $l : \mathcal{R}^{\mathrm{op}} \to \mathbf{Bool}$ by setting $l(x) = \mathtt{true}$ iff $l(x) \in L$. The result is monotone: $y \to x$ in $\mathcal{R}^{\mathrm{op}}$ means $x \to y$ in $\mathcal{R}$. If $l(x) = \mathtt{false}$, then $l(y) = \mathtt{false}$, because $x \notin L \Rightarrow y \notin L$ by the contrapositive of the lower set property. Similarly, if $l(y) = \mathtt{true}$, then also $l(x) = \mathtt{true}$. It is immediate that both operations described are inverse. Therefore, $\mathbf{Bool}^{\mathcal{R}^{\mathrm{op}}} \cong \mathsf{L}\mathcal{R}$.

Furthermore, if $\mathcal{R}$ has no infinite ascending chains, then each lower set corresponds uniquely to an antichain by taking its maximal elements. Since we order antichains by the order induced by their lower closures, we have $\mathsf{L}\mathcal{R} \cong \mathsf{A}\mathcal{R}$.

In summary, if $\mathcal{R}$ has no infinite ascending chains, we have the following

series of isomorphisms:

$$\begin{aligned}
\mathrm{Hom}(\mathcal{R}^{\mathrm{op}} \times \mathcal{F}, \mathbf{Bool}) &\cong \mathrm{Hom}(\mathcal{F} \times \mathcal{R}^{\mathrm{op}}, \mathbf{Bool}) && (1)\\
&\cong \mathrm{Hom}(\mathcal{F}, \mathbf{Bool}^{\mathcal{R}^{\mathrm{op}}}) && (2)\\
&\cong \mathrm{Hom}(\mathcal{F}, \mathsf{L}\mathcal{R}) && (3)\\
&\cong \mathrm{Hom}(\mathcal{F}, \mathsf{A}\mathcal{R}) && (4)
\end{aligned}$$

It follows that our feasibility relation Φ corresponds uniquely to a monotone function $h : \mathcal{F} \to \mathsf{A}\mathcal{R}$. To see what this function does, we follow the isos above. (1) sends Φ to its partial evaluations $f \mapsto \Phi(-, f)$. For each $f \in \mathcal{F}$, the function $\Phi(-, f)$ corresponds to the lower set

$$\Phi(-, f)^{-1}(\texttt{true}) = \{r \in \mathcal{R} \mid \Phi(r, f) = \texttt{true}\}.$$

Taking maximal elements of this gives an antichain of resources, which are the cheapest resources which are feasible. But this is precisely the query $f_{\max}$ described in section 2.1. Hence we have show that querying correspond to equivalence described by the isomorphisms above, provided $\mathcal{R}$ fulfils the ascending chain condition.

We can repeat the above by swapping the roles of $\mathcal{F}$ and $\mathcal{R}$. Observe that $\mathbf{Bool}^{\mathcal{F}} \cong (\mathsf{L}\mathcal{F})^{\mathrm{op}}$ as follows: If $h : \mathcal{F} \to \mathbf{Bool}$, then $h^{-1}(\texttt{false})$ is a lower set. If $h(y) = \texttt{false}$ and $x \to y$, then by monotonicity $h(x) \to h(y)$, so $x \in h^{-1}(\texttt{false})$. Moreover, if $g, h : \mathcal{F} \to \mathbf{Bool}$ with $g \Rightarrow h$, then for each $r \in \mathcal{F}$, we have $g(r) \to h(r)$. Hence, if $h(r) = \texttt{false}$, then $g(r) = \texttt{false}$, so $h^{-1}(\texttt{false}) \subseteq g^{-1}(\texttt{false})$. This is the opposite of the usual ordering on $\mathsf{L}\mathcal{F}$. The inverse operation is obtained by setting $l(x) = \texttt{false}$ iff x is in the lower set. Thus $\mathbf{Bool}^{\mathcal{F}} \cong (\mathsf{L}\mathcal{F})^{\mathrm{op}}$, as claimed.

We must now change up our isos slightly:. Assuming $\mathcal{F}$ has no infinite ascending chains:

$$\begin{aligned}
\mathrm{Hom}(\mathcal{R}^{\mathrm{op}} \times \mathcal{F}, \mathbf{Bool}) &\cong \mathrm{Hom}(\mathcal{R}^{\mathrm{op}}, \mathbf{Bool}^{\mathcal{F}}) && (5)\\
&\cong \mathrm{Hom}(\mathcal{R}^{\mathrm{op}}, (\mathsf{L}\mathcal{F})^{\mathrm{op}}) && (6)\\
&\cong \mathrm{Hom}(\mathcal{R}, \mathsf{L}\mathcal{F}) && (7)\\
&\cong \mathrm{Hom}(\mathcal{R}, \mathsf{A}\mathcal{F}) && (8)
\end{aligned}$$

The composite map sends $r \in \mathcal{R}$ to the antichain of the 'cheapest' functionalities in $\mathcal{F}$ that are still infeasible.

By swapping out lower sets for upper sets in the above, we should be able to get two more querying operation, provided the target poset fulfils a descending chain condition: One sends $f \in \mathcal{F}$ to the antichain of the most 'expensive' resources that still make the functionality infeasible. The other sends a resource to the most 'expensive' antichain of functionalities that makes the pairs feasible.

2.4 Bool-Posets

We have seen above that there is a tight relation between maps into **Bool** and lower sets, upper sets, and antichains. This leads up to consider such maps as the objects of a category.

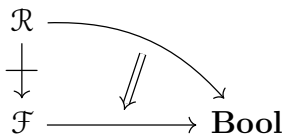
Definition 2.3. A ***Bool**-poset* is just a monotone map $\mathcal{P} \to \mathbf{Bool}$ from a poset $\mathcal{P}$ into **Bool**.

Next we define feasibility relations between **Bool**-posets.

Definition 2.4. A feasibility relation between **Bool**-posets $\rho : \mathcal{R} \to \mathbf{Bool}$ and $\varphi : \mathcal{F} \to \mathbf{Bool}$, is a feasibility realtion $\mathcal{R} \overset{\Phi}{\nrightarrow} \mathcal{F}$, where for all $r \in \mathcal{R}$ and $f \in \mathcal{F}$,

$$\Phi(r, f) = \texttt{true} \Rightarrow \rho(r) \leq \varphi(f).$$

We can express this condition in the following diagram which commutes up to a natural transformation:



Warning! Requiring the condition $\varphi(f) \leq \rho(r)$ leads to the restriction that resources above a certain threshold must be mapped bellow a certain functionality threshold, which is the opposite of what we want.

Lemma 2.2. Bool-*posets and their feasibility relationships form a category, denoted* $\mathbf{Pos}_{\mathbf{Bool}}$. *Composition is given by the usual composition of feasibility relations.*

Proof. If $\Phi : X \nrightarrow Y$, $\Psi : Y \nrightarrow Z$ are feasibility relations between **Bool**-posets $\xi : X \to \mathbf{Bool}$, $\upsilon : Y \to \mathbf{Bool}$ and $\zeta : Z \to \mathbf{Bool}$, then if $\Phi \fatsemi \Psi(x, z)$, the exists $y \in Y$ such that $\Phi(x, y)$ and $\Psi(y, z)$. Hence $\xi(x) \leq \upsilon(y)$ and $\upsilon(y) \leq \zeta(z)$, whence $\xi(x) \leq \zeta(z)$. This shows that composition is well-defined. Associativity is inherited from the composition of feasibility relations. Finally, there are identities given by the unit feasibility relation $\mathrm{id} : X \nrightarrow X$, with $\mathrm{id}(x, y) = \texttt{true}$ iff $x \to y \in X$. These satisfy the condition for being a morphism in $\mathbf{Pos}_{\mathbf{Bool}}$: If $\mathrm{id}(x, y) = \texttt{true}$, then $x \to y$, so $\xi(x) \leq \xi(y)$. Moreover, they are the identities for usual feasibility relations (see Fong / Spivak Lemma 4.19), so they also act that way here. ⌝⌞

The category $\mathbf{Pos}_{\mathbf{Bool}}$ has some interesting features, which we explore below:

Feasibility Relations are Objects Since a feasibility relation $\Phi : \mathcal{R}^{\mathrm{op}} \times \mathcal{F} \to \mathbf{Bool}$ is just a monotone map into **Bool**, it is an object of $\mathbf{Pos}_{\mathbf{Bool}}$. We can thus build feasibility relations between feasibility relations: If $\Psi : \mathcal{R}'^{\mathrm{op}} \times \mathcal{F}' \to \mathbf{Bool}$ is another feasibility relation, a second order feasibility relation $\Phi \overset{\Delta}{\nrightarrow} \Psi$ would be a monotone map

$$\Delta : (\mathcal{R}^{\mathrm{op}} \times \mathcal{F})^{\mathrm{op}} \times \mathcal{R}'^{\mathrm{op}} \times \mathcal{F}' \to \mathbf{Bool}$$

satisfying

$$\Delta((r, f), (r', f')) = \texttt{true} \Rightarrow \Phi(r, f) \leq \Psi(r', f').$$

Hence if $((r, f), (r', f'))$ is feasible, then $\Phi(r, f) \Rightarrow \Psi(r', f')$. Hence our association must preserve feasibility.

Limits and Colimits

Inclusion of Pos

Inclusion of DP

Projection to DP

The above form an adjunction

2.5 Queries in $\mathbf{Pos}_{\mathbf{Bool}}$

2.5.1 Generalized Querying

We can generalize the notion of query within $\mathbf{Pos}_{\mathbf{Bool}}$. We can interpret an object $\varphi : \mathcal{F} \to \mathbf{Bool}$ as representing a poset together with a chosen lower set (resp. upper set). This is done by considering the pre-image of `false` (resp. `true`). Conversely, choosing a lower set (resp. upper set) on a poset gives an object of $\mathbf{Pos}_{\mathbf{Bool}}$.

Using this interpretation, let $\mathcal{R}, \mathcal{F}$ be posets representing resources and functionalities, respectively. Choose lower sets $L_{\mathcal{R}} \subseteq \mathcal{R}$ and $L_{\mathcal{F}} \subseteq \mathcal{F}$. Recall that lower sets represent resources / functionalities that closed under being more 'expensive'. If the posets fulfil the ascending chain condition, we can think of the lower sets as thresholds, where we include anything more 'expensive' than the elements of some antichain.

Now observe that a morphism in $\mathbf{Pos}_{\mathbf{Bool}}$ is a feasibility relation $\mathcal{R} \overset{\Phi}{\nrightarrow} \mathcal{F}$, such that if (r, f) is feasible then $\rho(r) \leq \varphi(f)$, where $\varphi(f) = \texttt{false}$ iff $f \in L_{\mathcal{F}}$, and $\rho(r) = \texttt{false}$ iff $r \in L_{\mathcal{R}}$. Hence, if (r, f) is feasible and $f \in L_{\mathcal{F}}$, then $r \in L_{\mathcal{R}}$. Conversely, if $r \in \mathcal{R} \setminus L_{\mathcal{R}}$, then $f \in f \in \mathcal{F} \setminus L_{\mathcal{F}}$. In our threshold interpretation this means the following for a feasible pair (r, f): If f lies above (or on) our threshold for functionality, then r must lie above our threshold for resources. Conversely, if r lies below our resource threshold, then f lies below the functionality threshold.

Lets first think about varying $L_{\mathcal{R}}$. On one extreme if we let $L_{\mathcal{R}} = \mathcal{R}$, this would impose no restrictions whatsoever. On the other, if we choose $L_{\mathcal{R}} = \emptyset$, then only feasible pairs are allowed where f lies below the functionality threshold. Now lets vary $L_{\mathcal{F}}$. If $L_{\mathcal{F}} = \mathcal{F}$, then only feasible pairs with r above the resource threshold are allowed. If $L_{\mathcal{F}} = \emptyset$, we impose no restrictions.

Remark 2.2. It may seem like the variances are off here. However, suppose we had a condition that said, "if you are above a certain resource threshold, then you are above a certain funcionality threshold". This can't work, since if (r, f) is feasible and r is above the threshold, it may be that f is also above the threshold, but because of monotonicity (r, f') will also be feasible for any $f \to f'$. Such an f' could then lie below the threshold. So our threshold would have to be trivial in order to have any feasible pairs.

We can ask the following: Given some fixed feasibility relation and fixed lower set $L_{\mathcal{F}}$ in $\mathcal{F}$, i.e. some lower threshold for functionality, how low do we need to set the threshold in $\mathcal{R}$, i.e. how big do we need to make $L_{\mathcal{R}}$ until

we get a feasibility relation that can satisfy the condition for a morphism in $\mathbf{Pos}_{\mathbf{Bool}}$.

To see what this amounts to, lets take some feasibility relation Φ and fix $f \in \mathcal{F}$. We then set $L_{\mathcal{F}} = {\downarrow} f$. Our optimization procedure above would then yield a minimal lower set $L_{\mathcal{R}}$ in $\mathcal{R}$ such that if $\Phi(r, f)$ and $g \in {\downarrow} f$, then $r \in L_{\mathcal{R}}$. Assuming $\mathcal{R}$ has no infinite ascending chains, we can associate to $L_{\mathcal{R}}$ an antichain whose elements r are the 'cheapest' resources making (r, f) feasible: The minimality on $L_{\mathcal{R}}$ means that all elements r in $L_{\mathcal{R}}$ do in fact make (r, f) feasible, otherwise we could have excluded them to obtain a smaller lower set. The restriction on $L_{\mathcal{R}}$ means that we have found the cheapest resources for which f is feasible. If we take any element s that lies strictly below the antichain, (s, f) will not be feasible. For if (s, f) were feasible, then $s \in L_{\mathcal{R}}$.

In summary, we have shown that we can describe the optimization of section 2.1 in this setting. However, we can more generally query antichains of functionalities, rather than just single elements.

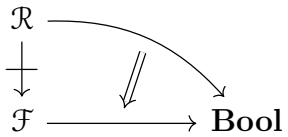
2.5.2 The Generalized Query Functor

Generalized querying as described in section 2.5.1 yields a contravariant functor $\mathbf{DP}_{\text{asc}} \to \mathbf{Pos}$, where $\mathbf{DP}_{\text{asc}}$ is the subcategory of $\mathbf{DP}$ where all posets satisfy the ascending chain condition. It sends a feasibility relation $\mathcal{R} \overset{\Phi}{\nrightarrow} \mathcal{F}$ to the monotone map $H_\Phi : \mathsf{A}\mathcal{F} \to \mathsf{A}\mathcal{R}$ which assigns each antichain in $\mathcal{F}$ the antichain of resources in $\mathcal{R}$ obtained by querying.

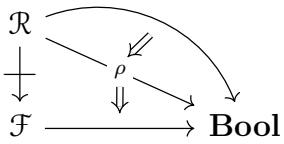
Functoriality ...

2.6 Queries as Universal Objects in $\mathbf{Pos}_{\mathbf{Bool}}$

Recall the diagram expressing the condition for morphisms in $\mathbf{Pos}_{\mathbf{Bool}}$.



We can ask whether for fixed $\mathcal{R} \overset{\Phi}{\nrightarrow} \mathcal{F}$ and fixed $\phi : \mathcal{F} \to \mathbf{Bool}$, there is a universal ρ: For any $\rho' : \mathcal{R} \to \mathbf{Bool}$ which makes Φ a morphism in $\mathbf{Pos}_{\mathbf{Bool}}$, we have $\rho'(r) \leq \rho(r)$ for all $r \in \mathcal{R}$. This means ρ is the maximal monotone map that makes Φ a morphism in $\mathbf{Pos}_{\mathbf{Bool}}$.



Lets see what this means. We can think about ρ and φ as a lower sets $L_\rho \subseteq \mathcal{R}$ and $L_\varphi \subseteq \mathcal{F}$ in and by taking the pre-images of `false`. Because ρ makes Φ a morphism in $\mathbf{Pos}_{\mathbf{Bool}}$, if (r, f) is feasible and $f \in L_\phi$, then $r \in L_\rho$. Moreover, the universality of ρ makes L_ρ the smallest lower set making Φ a morphism in $\mathbf{Pos}_{\mathbf{Bool}}$. This is exactly the result of the optimization in section 2.5.1. In summary, we have given a description of the process there in terms of a universal property.

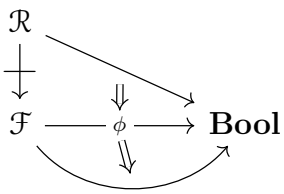
Lemma 2.3. *We can obtain the universal ρ by setting $\rho(r) = \bigwedge_{\{f \mid \Phi(r,f)\}} \varphi(f)$*

Proof. The result is monotone: If $r \to s$ and $\rho(s) = \texttt{false}$, then there is f with $\varphi(f) = \texttt{false}$ and $\Phi(s, f) = \texttt{true}$. But then $\Phi(r, f) = \texttt{true}$ by monotonicity, hence $\rho(f) = \texttt{false}$ as well. On the other hand if $\rho(r) = \texttt{true}$, then $\varphi(f) = \texttt{true}$ for all f satisfying $\Phi(r, f) = \texttt{true}$. If $\rho(s)$ were `false`, then there is a f with $\Phi(s, f) = \texttt{true}$ and $\varphi(f) = \texttt{false}$. But by monotonicity also $\Phi(r, f) = \texttt{true}$, a contradiction.

Finally, we show that this ρ is universal. First, suppose $\Phi(r, f) = \texttt{true}$. If $\varphi(f) = \texttt{false}$, then we have a `false` in the big wedge, so $\rho(r) =$ false. Hence, ρ makes Φ a morphism in $\mathbf{Pos}_{\mathbf{Bool}}$. Suppose we have any other ρ' making Φ a morphism. It is sufficient to show $\rho'(r) \Rightarrow \rho(r)$. Suppose $\rho'(r) = \texttt{true}$. If $\rho(r) = \texttt{false}$, then there would be f such that (r, f) is feasible and $\varphi(f)$ is `false`. But (r, f) feasible implies that $\texttt{true} = \rho'(r) \leq \varphi(r) = \texttt{false}$, a contradiction. Therefore $\rho(r) = \texttt{true}$ and we have proved universality. ⌐╦╜

2.6.1 Describing Other Queries

The other optimization problem we can describe in this way is shown in the following diagram



The univesal φ would satisfy: For any φ' making Φ a morphism in $\mathbf{Pos_{Bool}}$, we have $\phi \leq \phi'$. This makes the associated lower set L_ϕ the largest lower set making a given antichain of resources feasible.

3 Proslice $\mathcal{Q}$-Categories

The aim of this section is to generalize the construction given in [SD04]. In that paper, P_F-Sets are constructed based on the category **Rel** of sets and relations. That construction seems to work in general for a compact closed category, whose morphisms into the valuation object are partially ordered. This setting is generalized to the category $\mathbf{Prof}(\mathcal{Q})$ of $\mathcal{Q}$-profunctors, where $\mathcal{Q}$ is a commutative quantale. The valuation object can then be any $\mathcal{Q}$-enriched category, including $\mathcal{Q}$ itself.

3.1 Quantales

The basic definitions are taken from [MZ18]. The relation of quantales to linear logic is described in [Yet90].

Definition 3.1 (Quantale)**.** A *quantale* is a complete join semilattice with a monoid structure $(\circledast, k)$ where the multiplication distributes over arbitrary joins:

$$p \circledast \left(\bigvee_i q_i \right) = \bigvee_i p \circledast q_i \quad \text{and} \quad \left(\bigvee_i p_i \right) \circledast q = \bigvee_i p_i \circledast q$$

A *commutative quantale* is a quantale whose underlying monoid is commutative.

In the following we let $\mathcal{Q}$ denote a commutative quantale.

Definition 3.2 ($\mathcal{Q}$-enriched Category)**.** A $\mathcal{Q}$-enriched category $\mathcal{P}$ consist of:

- A set of *objects* $\mathrm{Ob}(\mathcal{P})$
- For each pair of objects, there is a *hom object* $\mathcal{P}(a, b) \in \mathcal{Q}$ satisfying:
  (i) $k \leq \mathcal{P}(a, a)$
  (ii) $\mathcal{P}(b, c) \circledast \mathcal{P}(a, b) \leq \mathcal{P}(a, c)$.

Definition 3.3 (Q-enriched Functor)**.** If $\mathcal{A}$ and $\mathcal{B}$ are Q-enriched categories, then a Q-*enriched functor* $F : \mathcal{A} \to \mathcal{B}$ consists of a function

$$F : \mathrm{Ob}(\mathcal{A}) \to \mathrm{Ob}(\mathcal{B})$$

satisfying

$$\mathcal{A}(a, b) \leq \mathcal{B}(Fa, Fb).$$

This gives us an ordinary category $\mathbf{Cat}(\mathcal{Q})$ whose objects are Q-enriched categories and whose morphisms are Q-enriched functors.

Because Q is thin, there can be at most one natural transformation between any two Q-functors.

Definition 3.4 (Q-enriched Natural Transformation)**.** Given parallel Q-enriched functors $F, G : \mathcal{A} \to \mathcal{B}$, the presence of a Q-*natural transformation* $\alpha : F \to G$ simply states that for all $a \in \mathcal{A}$ we have

$$k \leq \mathcal{B}(Fa, Ga).$$

Every commutative quantale is automatically closed:

Lemma 3.1. *Let* Q *be a commutative quantale. Then* $- \circledast b$ *has a right adjoint* $b \multimap -$ *given by*

$$b \multimap c := \bigvee_{a \circledast b \leq c} a.$$

Proof. We regard Q as a thin ordinary category. Then $- \circledast b$ is a function $\mathcal{Q} \times \mathcal{Q} \to \mathcal{Q}$. Moreover, if $a \leq a'$ and $b \leq b'$, then $a \vee a' = a'$ and $b \vee b' = b'$. Hence

$$a' \circledast b' = (a \vee a') \circledast (b \vee b') = (a \circledast b) \vee (a \circledast b') \vee (a' \circledast b) \vee (a' \circledast b').$$

In particular, $a \circledast b \leq a' \circledast b'$, showing $- \circledast b$ is a functor. Its right adjoint need to satisfy the natural iso

$$\mathcal{Q}(a \circledast b, c) \cong \mathcal{Q}(a, b \multimap c),$$

which reduces to the requirement that

$$a \circledast b \leq c \Leftrightarrow a \leq b \multimap c.$$

The left-to-right implication is immediate, as by assumption a appears in the join. For the right-to-left implication, suppose $a \leq \bigvee_{x \circledast b \leq c} x$. Then

$$a \circledast b \leq \left(\bigvee_{x \circledast b \leq c} x \right) \circledast b = \bigvee_{x \circledast b \leq c} x \circledast b \leq c,$$

because the join is the least upper bound. ⌝⌜

The previous lemma implies that we can regard $\mathcal{Q}$ as being enriched over itself by setting $\mathcal{Q}(a, b) = a \multimap b$. In particular, we can can consider $\mathcal{Q}$-functors into $\mathcal{Q}$.

Definition 3.5 ((Co)Presheaf)**.** Let $\mathcal{A}$ be a $\mathcal{Q}$-category. A *copresheaf* is a $\mathcal{Q}$-functor $F : \mathcal{A} \to \mathcal{Q}$. This means F must satisfy

$$\mathcal{A}(a, b) \leq F(a) \multimap F(b), \text{ or equivalently } \mathcal{A}(a, b) \circledast F(a) \leq F(b).$$

A *presheaf* is a $\mathcal{Q}$-functor $G : \mathcal{A}^{\mathrm{op}} \to \mathcal{Q}$. This means G must satisfy

$$\mathcal{A}(a, b) \leq F(b) \multimap F(a), \text{ or equivalently } \mathcal{A}(a, b) \circledast F(b) \leq F(a).$$

The monoidal structure of $\mathcal{Q}$ can be lifted to $\mathcal{Q}$-categories.

Definition 3.6 (Product Category)**.** Let $\mathcal{A}, \mathcal{B}$ be $\mathcal{Q}$-categories. Define the *product category* $\mathcal{A} \circledast \mathcal{B}$ to be the $\mathcal{Q}$-category with objects $\mathrm{Ob}(\mathcal{A}) \times \mathrm{Ob}(\mathcal{B})$ and hom objects

$$\mathcal{A} \circledast \mathcal{B}((a, b), (a', b')) := \mathcal{A}(a, a') \circledast \mathcal{B}(b, b').$$

Lemma 3.2. *The operation $- \circledast -$ defines a bifunctor on* $\mathbf{Cat}(\mathcal{Q})$.

Proof. Let $\alpha : A \to B$, $\gamma : C \to D$ be $\mathcal{Q}$-functors. Define $(\alpha \circledast \gamma)(a, b) := (\alpha(a), \gamma(b))$. To see that this is again a $\mathcal{Q}$-functors, we note that

$$A(a, a') \leq B(\alpha(a), \alpha(a')) \quad \text{and} \quad C(c, c') \leq D(\gamma(d), \gamma(d'))$$

Hence, by monotonicity of $\circledast$ (see proof of Lemma 3.1),

$$\begin{aligned}
(A \circledast C)(a, c)(a', c') &= A(a, a') \circledast C(c, c') \\
&\leq B(\alpha(a), \alpha(a')) \circledast D(\gamma(d), \gamma(d')) \\
&= (B \circledast D)(\alpha(a), \gamma(c))(\alpha(a'), \gamma(c')) \\
&= (B \circledast D)((\alpha \circledast \gamma)(a, c))((\alpha \circledast \gamma)(a', c')).
\end{aligned}$$

Functoriality is immediate:

$$(\mathrm{id}_A \circledast \mathrm{id}_C)(a,c) = (\mathrm{id}_A(a), \mathrm{id}_C(c)) = (a,c) = \mathrm{id}_{A\circledast C}(a,c)$$

$$((\alpha'\alpha)\circledast(\gamma'\gamma))(a,c) = (\alpha'\alpha(a), \gamma'\gamma(c)) = (\alpha'\circledast\gamma')(\alpha(a),\gamma(c)) = (\alpha'\circledast\gamma')\circ(\alpha\circledast\gamma)(a,c)$$

Hence $- \circledast -$ defines a bifunctor on $\mathbf{Cat}(\mathcal{Q})$. ⌐¬ L

Definition 3.7 (Opposite Category)**.** If $\mathcal{A}$ is a $\mathcal{Q}$-category, its *opposite category* $\mathcal{A}^{\mathrm{op}}$ has the same objects and hom objects $\mathcal{A}^{\mathrm{op}}(a,a') := \mathcal{A}(a',a)$

Definition 3.8 (Unit Category)**.** There is a *unit* $\mathcal{Q}$*-category* $\mathcal{I}$ consisting of a single object and the quantale unit k as its single hom object.

Definition 3.9 (Profunctor)**.** Let $\mathcal{A}, \mathcal{B}$ be $\mathcal{Q}$-categories. A *profunctor* $R : \mathcal{A} \nrightarrow \mathcal{B}$ is a $\mathcal{Q}$-functor $R : \mathcal{A}^{\mathrm{op}} \circledast \mathcal{B} \to \mathcal{Q}$. Unravelling this definition shows that R must satisfy

$$\mathcal{A}(a',a) \circledast R(a,b) \circledast \mathcal{B}(b,b') \leq R(a',b').$$

Given profunctors $R : \mathcal{A} \nrightarrow \mathcal{B}$ and $S : \mathcal{B} \nrightarrow \mathcal{C}$, we can define their composite $S \circ R : \mathcal{A} \nrightarrow \mathcal{C}$ to be

$$(S \circ R)(a,c) = \bigvee_b R(a,b) \circledast S(b,c).$$

This composition is associative and has an identity given by $1_{\mathcal{A}}(a,a') := \mathcal{A}(a,a')$. Therefore $\mathcal{Q}$-profunctors form a category $\mathbf{Prof}(\mathcal{Q})$.

Lemma 3.3. *The operation* $- \circledast -$ *defines a bifunctor* $\mathbf{Prof}(\mathcal{Q})$.

Proof. We need to specify where the tensor takes morphisms. Let $\Phi : A \nrightarrow B$, $\Psi : C \nrightarrow D$ be morphisms between $\mathcal{Q}$-categories A, B, C, and D. Define $\Phi \circledast \Psi : A \circledast C \nrightarrow B \circledast D$ by

$$\Phi \circledast \Psi((a,c),(b,d)) := \Phi(a,b) \circledast \Psi(c,d).$$

To see that this specifies a $\mathcal{Q}$-profunctor we write down the conditions satisfied by Φ and Ψ

$$A(a',a) \circledast \Phi(a,b) \circledast B(b,b') \leq \Phi(a',b')$$
$$C(c',c) \circledast \Psi(c,d) \circledast D(d,d') \leq \Psi(c',d')$$

Because the operation $\circledast$ is monotone (see proof of Lemma 3.1), this implies

$$A(a',a) \circledast C(c',c) \circledast \Phi(a,b) \circledast \Psi(c,d) \circledast B(b,b') \circledast D(d,d') \leq \Phi(a',b') \circledast \Psi(c',d')$$

Using the definition of the product, the above is equivalent to

$$\begin{aligned}(A \circledast C)((a',c'),(a,c)) \circledast (\Phi \circledast \Psi)((a,c),(b,d)) \circledast (B \circledast D)((b,d),(b',d')) \\ \leq (\Phi \circledast \Psi)((a',c'),(b',d')),\end{aligned}$$

which is exactly the requirement we need. Next we turn to the functoriality requirements. It is immediate form the definitions that

$$\begin{aligned}(\mathrm{id}_A \circledast \mathrm{id}_B)((a,b),(a',b')) &= \mathrm{id}_A(a,a') \circledast \mathrm{id}_B(b,b') \\ &= A(a,a') \circledast B(b,b') \\ &= (A \circledast B)((a,b),(a',b')) \\ &= \mathrm{id}_{A \circledast B}((a,b),(a',b'))\end{aligned}$$

Moreover, if in addition $\Phi' : B \nrightarrow X$ and $\Phi' : D \nrightarrow Y$, then

$$\begin{aligned}(\Phi' \circ \Phi) \circledast (\Psi' \circ \Psi)((a,c),(x,y)) &= (\Phi' \circ \Phi)(a,x) \circledast (\Psi' \circ \Psi)(c,y) \\ &= \left(\bigvee_{b \in B} \Phi(a,b) \circledast \Phi'(b,x) \right) \circledast \left(\bigvee_{d \in D} \Psi(c,d) \circledast \Psi'(d,y) \right) \\ &= \bigvee_{b \in B} \bigvee_{d \in D} \Phi(a,b) \circledast \Phi'(b,x) \circledast \Psi(c,d) \circledast \Psi'(d,y) \\ &= \bigvee_{(b,d) \in B \circledast D} (\Phi \circledast \Psi)((a,c),(b,d)) \circledast (\Phi' \circledast \Psi')((b,d),(x,y)) \\ &= (\Phi \circledast \Psi) \circ (\Phi' \circledast \Psi')((a,c),(x,y))\end{aligned}$$

Hence $- \circledast -$ defines a bifunctor on $\mathbf{Prof}(\mathcal{Q})$. ⌐¬

Proposition 3.1. *The category* $\mathbf{Prof}(\mathcal{Q})$ *is symmetric monoidal under* $- \circledast -$ *with unit* $\mathcal{I}$*. Moreover it has a compact closed where each object is its own dual.*

Proof. . . . ⌐¬

3.2 Proslice Construction

We recall that in in the case of the **Bool**-posets of section 2.4, we defined morphisms between $\alpha : A \to \mathbf{Bool}$ and $\beta : B \to \mathbf{Bool}$ (see 2.4) as **Bool**-profunctors $\Phi : A \nrightarrow B$ such that if $\Phi(a, b) = \texttt{true}$, then $\alpha(a) \leq \beta(b)$. We can make this the following condition in **Bool**:

$$\Phi(a, b) \leq \alpha(a) \multimap \beta(b) = \mathbf{Bool}(\alpha(a), \beta(b)).$$

This allows us to generalize the condition to any $\mathcal{Q}$-category. Furthermore, we twist the construction by an endofunctor for added flexibility. Together this yields:

Definition 3.10 (Proslice Category)**.** Let $\mathcal{Q}$ be a quantale, $E : \mathbf{Prof}(\mathcal{Q}) \to \mathbf{Prof}(\mathcal{Q})$ be an endofunctor, and $\mathcal{V}$ a $\mathcal{Q}$-category. We define the *proslice category* $\mathbf{Prof}(\mathcal{Q}) \wr \mathcal{V}$ *over* $\mathcal{V}$ to be the category with

(i) objects are $\mathcal{Q}$-functors $EA \to \mathcal{V}$, where $A \in \mathbf{Prof}(\mathcal{Q})$.

(ii) For objects $\alpha : EA \to \mathcal{V}$, $\beta : EB \to \mathcal{V}$, morphisms are profunctors $\Phi : A \nrightarrow B$, satisfying

$$(E\Phi)(a, b) \leq \mathcal{V}(\alpha(a), \beta(b)).$$

Proof. Since E is an endofunctor and $\mathbf{Prof}(\mathcal{Q})$ a category, it suffices to check that the proslice construction is closed under identities and composition. If $\alpha(a) : EA \to \mathcal{V}$ is a $\mathcal{Q}$-functor, then it satisfies $EA(a, b) \leq \mathcal{V}(\alpha(a), \beta(b))$. But $E\ \mathrm{Id}_A(a, b) = EA(a, b)$, so the identity profunctor satisfies the condition to be a proslice morphism. Finally, if $A \overset{\Phi}{\nrightarrow} B$ and $B \overset{\Psi}{\nrightarrow} C$ are proslice morphisms, then $E\Phi(a, b) \leq \mathcal{V}(\alpha(a), \beta(b))$ and $E\Psi(b, c) \leq \mathcal{V}(\beta(b), \gamma(c))$. Therefore,

$$\begin{aligned}
E(\Psi \circ \Phi)(a, c) &= (E\Psi \circ E\Phi)(a, c) && \text{(functoriality)} \\
&= \bigvee_b E\Phi(a, b) \circledast E\Psi(b, c) && \text{(definition)} \\
&\leq \bigvee_b \mathcal{V}(\alpha(a), \beta(b)) \circledast \mathcal{V}(\beta(b), \gamma(c)) && \text{(proslice morph.)} \\
&\leq \bigvee_b \mathcal{V}(\alpha(a), \gamma(c)) && \text{(comp. cond.)} \\
&= \mathcal{V}(\alpha(a), \gamma(c)).
\end{aligned}$$

Hence the composite of two proslice morphisms is again proslice. ⌝⌜⌞⌟

3.2.1 Relation to Profunctor Collage

3.2.2 Relation to Order on Profunctors

3.3 Properties of $\mathbf{Prof}(\mathcal{Q}) \wr \mathcal{V}$

3.3.1 Tensor product

For the following we will assume that $\mathcal{V}$ is a monoidal $\mathcal{Q}$-category.

Definition 3.11 (Monoidal $\mathcal{Q}$-category)**.** A strict symmetric *monoidal $\mathcal{Q}$-category* is a $\mathcal{Q}$-category $\mathcal{M}$ is a monoid object in $\mathbf{Cat}(\mathcal{Q})$. That is, $\mathcal{M}$ comes equipped with a multiplication $\mathcal{Q}$-bifunctor $\otimes : \mathcal{M} \circledast \mathcal{M} \to \mathcal{M}$ and a unit $\mathcal{Q}$-functor $\mu : \mathcal{I} \to \mathcal{M}$ making the following diagrams commute:

$$\begin{array}{ccc} \mathcal{M} \circledast \mathcal{M} \circledast \mathcal{M} & \xrightarrow{\mathcal{M}\circledast\otimes} & \mathcal{M} \circledast \mathcal{M} \\ \downarrow{\scriptstyle \otimes\circledast\mathcal{M}} & & \downarrow{\scriptstyle \otimes} \\ \mathcal{M} \circledast \mathcal{M} & \xrightarrow{\otimes} & \mathcal{M} \end{array} \qquad \begin{array}{ccccc} \mathcal{M} \circledast \mathcal{I} & \xrightarrow{\mathcal{M}\circledast\mu} & \mathcal{M} \circledast \mathcal{M} & \xleftarrow{\mu\circledast\mathcal{M}} & \mathcal{I} \circledast \mathcal{M} \\ & \searrow^{\sim} & & \swarrow_{\sim} & \\ & & \mathcal{M} & & \end{array}$$

Lemma 3.4. *If $(\mathcal{V}, \otimes, \mu)$ is a monoidal $\mathcal{Q}$-category, then $\otimes$ is monotone on hom-objects:*

$$\mathcal{V}(v, v') \circledast \mathcal{V}(w, w') \leq \mathcal{V}(v \otimes w, v' \otimes w')$$

Proof. This is simply because $\otimes$ is a $\mathcal{Q}$-functor. ⌐

Definition 3.12 (Oplax Functor)**.** ...

Proposition 3.2. *Let $(\mathcal{Q}, \circledast, k)$ be commutative quantale, and $(\mathcal{V}, \otimes, \mu)$ a monoidal $\mathcal{Q}$-category. Suppose $E : \mathbf{Prof}(\mathcal{Q}) \to \mathbf{Prof}(\mathcal{Q})$ is the extension of an oplax monoidal functor $\tilde{E}$ on $\mathbf{Cat}(\mathcal{Q})$ with natural transformation $\vartheta : E(- \circledast -) \to E(-) \circledast E(-)$, such that $E(\Phi \circledast \Psi)(x, y) \leq (E(\Phi) \circledast E(\Psi))(\vartheta_{A,B}(x), \vartheta_{C,D}(y))$ for all $\Phi, \Psi \in \mathbf{Prof}(\mathcal{Q})$ (looks like it would follow from naturality of ϑ). Then $\mathbf{Prof}(\mathcal{Q}) \wr \mathcal{V}$ is a symmetric monoidal category where the tensor of $\alpha : EA \to \mathcal{V}$ and $\beta : EB \to \mathcal{V}$ is given by*

$$E(A \circledast B) \overset{\vartheta_{A,B}}{\to} E(A) \circledast E(B) \overset{\alpha\circledast\beta}{\to} \mathcal{V} \circledast \mathcal{V} \overset{\otimes}{\to} \mathcal{V}.$$

Written out as a composition, $\alpha \otimes \beta := \alpha\pi_1\vartheta_{A,B} \otimes \beta\pi_2\vartheta_{A,B}$.

The unit ι for the tensor is given by the object

$$E\mathcal{I} \overset{\vartheta_0}{\to} \mathcal{I} \overset{\mu}{\to} \mathcal{V},$$

that is, $\iota = \mu\vartheta_0$. The forgetful functor $\mathbf{Prof}(\mathcal{Q}) \wr \mathcal{V} \to \mathbf{Prof}(\mathcal{Q})$ preserves this tensor strictly.

Proof in progress... Let $\alpha : EA \to \mathcal{V}, \beta : EB \to \mathcal{V}, \gamma : EC \to \mathcal{V}$, and $\delta : ED \to \mathcal{V}$ be objects in $\mathbf{Prof}(\mathcal{Q}) \wr \mathcal{V}$. Moreover, let $\Phi : A \nrightarrow C$ and $\Psi : B \nrightarrow D$ be proslice morphisms. We first show that the proposed tensor product is indeed a bifunctor. We observe that because Φ and Ψ are proslice morphisms we have

$$E\Phi(a,c) \leq \mathcal{V}(\alpha(a), \gamma(c)) \quad \text{and} \quad E\Psi(b,d) \leq \mathcal{V}(\beta(b), \delta(d)).$$

Hence,

$$\begin{aligned} E(\Phi \circledast \Psi)(x,y) &\leq (E\Phi \circledast E\Psi)(\vartheta_{A,B}(x), \vartheta_{C,D}(y)) \\ &= E\Phi(\pi_1\vartheta_{A,B}(x), \pi_1\vartheta_{C,D}(y)) \circledast E\Psi(\pi_2\vartheta_{A,B}(x), \pi_2\vartheta_{C,D}(y)) \\ &\leq \mathcal{V}(\alpha\pi_1\vartheta_{A,B}(x), \gamma\pi_1\vartheta_{C,D}(y)) \circledast \mathcal{V}(\beta\pi_2\vartheta_{A,B}(x), \delta\pi_2\vartheta_{C,D}(y)) \\ &\leq \mathcal{V}(\alpha\pi_1\vartheta_{A,B}(x) \otimes \beta\pi_2\vartheta_{A,B}(x), \gamma\pi_1\vartheta_{C,D}(y) \otimes \delta\pi_2\vartheta_{C,D}(y)) \\ &= \mathcal{V}(\alpha \otimes \beta(x), \gamma \otimes \delta(y)) \end{aligned}$$

showing that the image of proslice morphisms under the tensor is again proslice.

Functoriality of the tensor follows from the functoriality of E and $- \circledast -$ of $\mathbf{Prof}(\mathcal{Q})$:

$$E(\mathrm{id}_A \circledast \mathrm{id}_B) = E(\mathrm{id}_{A \circledast B}) = \mathrm{id}_{E(A \circledast B)}$$

$$E((f \circ f') \circledast (g \circ g')) = E((f \circledast g) \circ (f' \circledast g')) = E(f \circledast g) \circ E(f' \circledast g').$$

It remains to define the required natural isomorphisms and check the axioms for a symmetric monoidal category. We set

$$\mathit{assoc} : E((A \circledast B) \circledast C) \cong E(A \circledast (B \circledast C))$$

to be the image under E of the usual natural iso for the product of categories. Similarly,

$$\mathit{right} : E(A \circledast \mathcal{I}) \cong E(A), \mathit{left} : E(\mathcal{I} \circledast A) \cong E(A) \text{ and } \mathit{swap} : E(A \circledast B) \cong E(B \circledast A)$$

are also given as images of the usual natural isos in the symmetric monoidal category $\mathbf{Prof}(\mathcal{Q})$. As a consequence, the axioms are automatically fulfilled, since the requisite diagrams are the images of commuting diagrams in $\mathbf{Prof}(\mathcal{Q})$. The final thing to check is that all the natural isos are in fact proslice morphisms.

Claim about forgetful functor.

3.4 Proslice Bool-Categories

We attempt to first develop the theory in the simplified setting of **Bool**-enriched categories. For the following denote **Bool** $=: \mathbb{B}$.

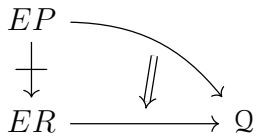
Definition 3.13 (Proslice **Bool**-categories)**.** Let $\mathcal{Q}$ be a commutative quantale, and $E : \mathbf{Prof}(\mathbb{B}) \to \mathbf{Prof}(\mathbb{B})$ an endofunctor. The *proslice category* $\mathbf{Prof}(\mathbb{B}) \wr_E \mathcal{Q}$ consists of:

(i) objects are monotone maps $\varphi : EP \to \mathcal{Q}$

(ii) morphisms between $\varphi : EP \to \mathcal{Q}$ and $\rho : ER \to \mathcal{Q}$ are feasibility relations $\Phi : P \nrightarrow R$ satisfying the *proslice condition*:

$$E\Phi(a, b) = \texttt{true} \Rightarrow \varphi(a) \leq \rho(b).$$

Diagrammatically:



We can define an order structure on the objects of the proslice category. Since we want the definition to cover both boolean profunctors and monotone maps, we regard both as special types of relations, and identify a relation $A \nrightarrow B$ with an arbitrary function $A \times B \to \mathbb{B}$.

Definition 3.14. Let $\Phi : A \nrightarrow B$ and $\Psi : A \nrightarrow B$ be arbitrary relations. Set $\Phi \leq \Psi$ iff for all $a \in A$ and $b, b' \in B$ we have that $\Phi(a, b) = \texttt{true}$ and $\Psi(a, b') = \texttt{true}$ imply $b \leq b'$. In particular, this order restricts to the pointwise order on monotone maps and hence on objects of the proslice category.

We can characterize the proslice condition in terms of the aforementioned order.

Lemma 3.5. *Let $\varphi : EP \to \mathcal{Q}$ and $\rho : ER \to \mathcal{Q}$ be proslice objects. A feasibility relation $\Phi : P \nrightarrow R$ satisfies the proslice condition if and only if $\varphi \leq \rho \circ E\Phi$.*

Proof. Suppose Φ is proslice. If $\varphi(a,b) = \texttt{true}$ and $\rho \circ E\Phi(a,b') = \texttt{true}$, then $\varphi(a) = b$ and there is $r \in ER$ such that $\rho(r) = b'$ and $\Phi(a,r) = \texttt{true}$. By the proslice condition $b = \varphi(a) \leq \rho(r) = b'$.

Conversely, suppose that $\varphi \leq \rho \circ E\Phi$ and $E\Phi(a,b) = \texttt{true}$. Because ρ is a function, we have $\rho(b,c) = \texttt{true}$ for $c := \rho(b)$. Then $\rho \circ E\Phi(a,c) = \texttt{true}$. Now for $\varphi(a) = c'$ we have $\phi(a,c') = \texttt{true}$, so $\phi(a) = c' \leq c = \rho(b)$ by assumption. Hence Φ is proslice. ⊏⊐

This characterization can be used to obtain some lemmas for diagram pasting. In short, pre- and post-composing with monotone maps preserved the order.

Lemma 3.6 (Pre-/Post-composition)**.** *Let $S, S' : B \nrightarrow C$ be feasibility relations and $R : A \to B$ $T : C \to D$ monotone maps. If $S \leq S'$, then*

(i) $S \circ R \leq S' \circ R$,

(ii) $T \circ S \leq T \circ S'$.

Proof. Suppose $S \leq S'$. For *(i)* assume that $S \circ R(a,c) = \texttt{true}$ and $S' \circ R(a,c') = \texttt{true}$. There are $b, b' \in B$ satisfying $R(a) = b$, $S(b,c) = \texttt{true}$ and $R(a) = b'$, $S'(b',c') = \texttt{true}$. But because R is a function $b = b'$, so we in fact have $S(b,c) = \texttt{true}$ and $S'(b,c') = \texttt{true}$, which implies $c \leq c'$ since $S \leq S'$.

For *(ii)* we assume that $T \circ S(b,d) = \texttt{true}$ and $T \circ S'(b,d') = \texttt{true}$. There are $c, c' \in C$ such that $T(c) = d$, $T(c') = d'$ and $S(b,c) = \texttt{true}$, $S'(b,c') = \texttt{true}$. Since $S \leq S'$, we know $c \leq c'$. Therefore by monotonicity of T we have $d = T(c) \leq T(c') = d'$. ⊏⊐

The order also behaves well with respect to the cartesian product on $\mathbf{Prof}(\mathbb{B})$.

Lemma 3.7 (Product)**.** *Let $\alpha, \alpha' : A \nrightarrow A'$, $\beta, \beta' : B \nrightarrow B'$ be feasibility relations. If $\alpha \leq \alpha'$ and $\beta \leq \beta'$, then $\alpha \times \beta \leq \alpha' \times \beta'$.*

Proof. Assume $\alpha \times \beta((a,b),(x,y)) = \texttt{true}$ and $\alpha' \times \beta'((a,b),(x',y')) = \texttt{true}$. By definition of the cartesian product of profunctors this means $\alpha(a,x) = \beta(b,y) = \alpha'(a,x') = \beta'(b,y') = \texttt{true}$. Using our hypotheses that $\alpha \leq \alpha'$ and $\beta \leq \beta'$, we obtain $x \leq x'$ and $y \leq y'$. Hence $(x,y) \leq (x',y')$, as desired. ⊏⊐

In the following we will need to choose the endofunctor on the proslice category to have a special form. We want there to be natural transformations $\vartheta : E(- \times -) \to E(-) \times E(-)$ and $E\mathbb{1} \to \mathbb{1}$ whose components are monotone maps. To make the definition we observe that both $\mathbf{Cat}(\mathcal{Q})$ and $\mathbf{Prof}(\mathcal{Q})$ embed into $\mathbf{Rel}_{\mathbf{Pos}}$, the category whose objects are posets and whose morphisms are arbitrary relations. Hence, we can formulate a mixed naturality condition there.

Definition 3.15. An endofunctor E on $\mathbf{Prof}(\mathbb{B})$ is called *slicing*, if it is oplax symmetric monoidal with respect to the cartesian product, and the components of the natural transformations are monotone functions. More precisely, for any two posets $A, B \in \mathbf{Prof}(\mathbb{B})$ we want a monotone map $\vartheta_{A,B} : E(A \times B) \to E(A) \times E(B)$ in $\mathbf{Cat}(\mathbb{B}) = \mathbf{Pos}$ that makes the following diagram commute in $\mathbf{Rel}_{\mathbf{Pos}}$:

$$\begin{array}{ccc} E(A \times B) & \xrightarrow{\vartheta_{A,B}} & E(A) \times E(B) \\ \downarrow{\scriptstyle E(\Phi \times \Psi)} & & \downarrow{\scriptstyle E\Phi \times E\Psi} \\ E(A' \times B') & \xrightarrow{\vartheta_{A',B'}} & E(A') \times E(B') \end{array}$$

We also require a monotone map $\vartheta_0 : E\mathbb{1} \to \mathbb{1}$.

Theorem 3.1 (Transfer)**.** *Suppose E is slicing. Then any monotone n-ary operation on $\mathcal{Q}$ induces a functor $\bullet : (\mathbf{Prof}(\mathbb{B}) \wr_E \mathcal{Q})^{\times n} \to \mathbf{Prof}(\mathbb{B}) \wr_E \mathcal{Q}$ as follows:*

Objects *$\alpha_i : EA_i \to \mathcal{Q}$ are mapped to the composite*

$$E(A_1 \times \ldots \times A_n) \xrightarrow{\vartheta_{A_1,\ldots,A_n}} E(A_1) \times \ldots \times E(A_n) \xrightarrow{\alpha_1 \times \ldots \times \alpha_n} \mathcal{Q}^{\times n} \xrightarrow{\bullet} \mathcal{Q}$$

where $\vartheta_{A_1,\ldots,A_n}$ denotes the iterated application of the appropriate components of the natural transformation ϑ.

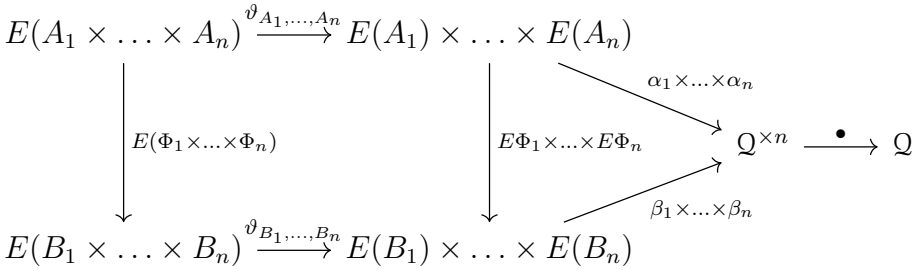
Morphisms *$E\Phi_i : EA_i \nrightarrow EB_i$ are mapped to $E(\Phi_1 \times \ldots \times \Phi_n)$.*

Proof. Let $\bullet : \mathcal{Q}^{\times n} \to \mathcal{Q}$ be the monotone n-ary operation in question. Suppose $\alpha_i : EA_i \to \mathcal{Q}$, $\beta_i : EB_i \to \mathcal{Q}$ are proslice objects, and $\Phi_i : A_i \nrightarrow B_i$ proslice morphisms, for $1 \leq i \leq n$. The proslice condition, $\alpha_i \leq \beta_i \circ E\Phi_i$

implies $\alpha_1 \times \ldots \times \alpha_n \leq \beta_1 \circ E\Phi_1 \times \ldots \times \beta_n \circ E\Phi_n$ by lemma 3.7. Because $\times$ is a functor,

$$\beta_1 \circ E\Phi_1 \times \ldots \times \beta_n \circ E\Phi_n = (\beta_1 \times \ldots \times \beta_n) \circ (E\Phi_1 \times \ldots \times E\Phi_n).$$

By appending the relevant iterated components of the natural transformation ϑ, we get the following diagram, where the left square commutes by naturality.



By using pre-composition lemma 3.6 and the commutativity of the square, we obtain

$$\begin{aligned}(\alpha_1 \times \ldots \times \alpha_n) \circ \vartheta_{A_1,\ldots,A_n} &\leq (\beta_1 \times \ldots \times \beta_n) \circ (E\Phi_1 \times \ldots \times E\Phi_n) \circ \vartheta_{A_1,\ldots,A_n} \\ &= (\beta_1 \times \ldots \times \beta_n) \circ \vartheta_{B_1,\ldots,B_n} \circ E(\Phi_1 \times \ldots \times \Phi_n).\end{aligned}$$

By post-composition, again using lemma 3.6,

$$\bullet[(\alpha_1 \times \ldots \times \alpha_n) \circ \vartheta_{A_1,\ldots,A_n}] \leq \bullet[(\beta_1 \times \ldots \times \beta_n) \circ \vartheta_{B_1,\ldots,B_n} \circ E(\Phi_1 \times \ldots \times \Phi_n)].$$

Therefore, $E(\Phi_1 \times \ldots \times \Phi_n)$ is proslice with respect to the induced operation. Finally, functoriality follows from that of $\times$ and E:

$$\begin{aligned}E(\Psi_1\Phi_1 \times \ldots \times \Psi_n\Phi_n) &= E((\Psi_1 \times \ldots \times \Psi_n) \circ (\Phi_1 \times \ldots \times \Phi_n)) \\ &= E(\Psi_1 \times \ldots \times \Psi_n) \circ E(\Phi_1 \times \ldots \times \Phi_n)\end{aligned}$$

and

$$E(\mathrm{id}_{A_1} \times \ldots \times \mathrm{id}_{A_n}) = E(\mathrm{id}_{A_1 \times \ldots \times A_n}) = \mathrm{id}_{E(A_1 \times \ldots \times A_n)}.$$

Remark 3.1. Observe that the above proof also works for any monoidal product $\otimes$ on $\mathbf{Prof}(\mathbb{B})$ and operation $\mathcal{Q}^{\otimes n} \to \mathcal{Q}$.

Our goal is now to transfer the structure of the quantale $\mathcal{Q}$ (which we will eventually choose to be *-autonomous) to the proslice category $\mathbf{Prof}(\mathbb{B})\wr_E \mathcal{Q}$. The transfer theorem 3.1 tells us that we will get appropriate functors on the proslice category for each operation on $\mathcal{Q}$. However, we will need to check that the induced operations also satisfy the requisite properties. We start with the monoidal product.

3.4.1 Symmetric Monoidal Structure

Proposition 3.3. *Let $(\mathcal{Q}, \otimes, k)$ be a commutative quantale and E slicing. Then $\mathbf{Prof}(\mathbb{B}) \wr_E \mathcal{Q}$ is a symmetric monoidal category with respect to the bifunctor induced by $\otimes$.*

Proof. By the transfer theorem 3.1, we know that we have a bifunctor $\otimes$ on $\mathbf{Prof}(\mathbb{B}) \wr_E \mathcal{Q}$. We need to show that this satisfies the axioms for a SMC. The required natural isomorphisms will be inherited from those that turn the cartesian product into a symmetric monoial structure on $\mathbf{Prof}(\mathbb{B})$. These are

$$\mathit{assoc} : (A \times B) \times C \to A \times (B \times C),$$

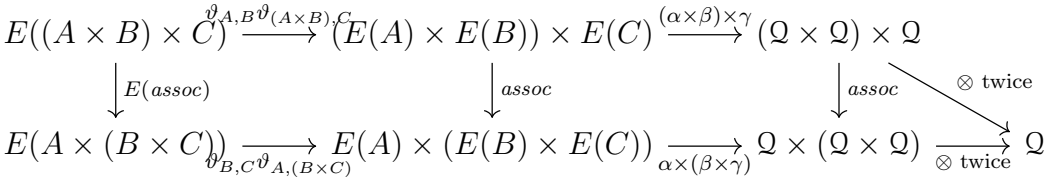
$$\mathit{left} : \mathbb{1} \times A \to A,$$

$$\mathit{right} : A \times \mathbb{1} \to A,$$

$$\mathit{swap} : A \times B \to B \times A.$$

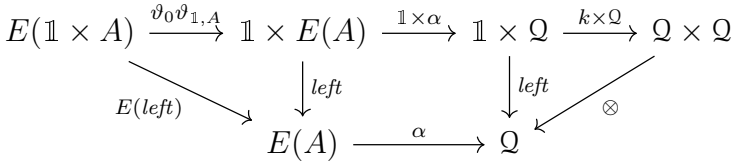
which are the appropriate profunctors induced by the corresponding functions in **Set**. For the proslice category, we take the images of the above natural isomorphisms under the functor E. This process preserves naturality and the commutativity of the requisite diagrams, since morphisms in the proslice category are just special morphisms in $\mathbf{Prof}(\mathbb{B})$. The only thing to check is that the compoents of our natural transformations actually lie in the proslice category, that is that they fulfil the proslice condition. This can be seen from the following commutative diagrams, where the top and bottom paths are the proslice objects obtained by the induced operation. The fact that they commute means that we in particular have the inequality of relations that characterizes proslice morphisms (see lemma 3.5).

We start with associativity:



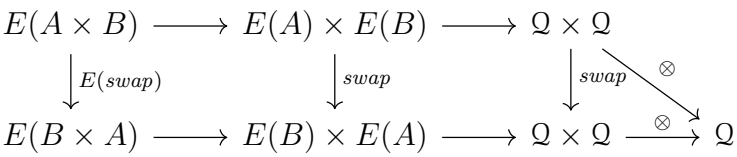
The leftmost square commutes by the associativity requirement for the oplax monoidal functor E, the middle square commutes by the naturality of *assoc*, and the rightmost triangle commutes because it is the image of the associativity diagram for Ω in **Pos**.

The diagram for the left unitor is:



The left triangle communtes by the axioms for an oplax monoidal functor, the middle square by naturality of *left*, and the right triangle because of the unit law for Ω. The diagram for the right unitor is analogous.

Finally, for commutativity we have



where the left square commutes because E is symmetric oplax monoidal, the middle by naturality of *swap*, and the right triangle by commutativity of Ω. ⌐╦╚

4 Bundles

4.1 The Setting

For the moment we will forget about resources and functionalities and instead just consider a preorder of things. An arrow $a \to b$ expresses the fact that if I have a, then I can obtain b from it. We imagine this as a process of instant conversion.

In this setting we can imagine bundling things together. Such a bundle unites a collection of things to a whole. The type of bundle dictates how we are able to interact with the things therein. These interactions are characterized by what things we can obtain using the bundles and from what things we can obtain a given bundle.

We will enforce resource conservation on the bundles, meaning that we can obtain each item occurring in the bundle at most once. Conversely, if the bundle make some resources available, we are stuck with them and can't just throw them away.

4.2 Agnostic Choice

To start with we will assume our bundles only track availability and unavailability of things. They will be agnostic to temporal, causal, spacial, or similar features. The bundles can nonetheless create some forms of dependency between the things.

4.2.1 2-Bundles

Here are all the bundles of two things a, b I can think of in this case:

Both(a,b) You have both a, b at the same time and can use both freely in any order.

Either(a,b) You can choose between either having a or having b.

Maybe(a,b) You have either a or b, but you don't know in advance which one. When using the bundle, it will give you either a or b.

BombL(a,b) Using a will destroy b, but not vice versa. This is equivalent to $\mathsf{Either}(a, \mathsf{Both}(a, b))$.

We could be tempted to also introduce an ordered pair $\mathsf{Before}(a, b)$ expressing the fact that we have both a, b, but in a set order (temporal, spacial, causal). Our assumption that the bundles are agnostic to these features implies $\mathsf{Before}(a, b) = \mathsf{Before}(b, a)$, regardless of how we interpret the ordering. Note that in any case

$$\mathsf{Both}(a, b) = \mathsf{Either}(\mathsf{Before}(a, b), \mathsf{Before}(b, a))).$$

Observe also that asymmetric dependence with respect to availability is expressed by $\mathsf{BombL}(a, b)$ and can be expressed in terms of Both and Either.

Finally, note we distinguish two modes of choice, free and forced. This corresponds to two agents. In principle, we could establish more complex forms of choice by considering agents with (possibly interrelated) knowledge pools. Our situation corresponds to two agents (me and my opponent) with no shared knowledge. Specifically, $\mathsf{Either}(a, b) \to \mathsf{Maybe}(a, b)$, from my perspective, but not the other way around. From my opponents perspective, the situation is reversed.

4.2.2 n-Bundles

Lets turn to bundles of higher arity. For example there could be a general bundle containing n things, some of which you can choose freely, and some of which will be chosen for you. We can express this using the elementary n-bundles $\mathsf{Either}(a_1, \ldots, a_n)$, expressing a single free choice from n elements and $\mathsf{Maybe}(a_1, \ldots, a_n)$, expressing a single forced choice from n elements.

For example, a bundle of 3 things with a forced choice followed by a free choice is given by

$$\mathsf{Maybe}(\mathsf{Either}(\mathsf{Both}(a, b), \mathsf{Both}(a, c)), \mathsf{Either}(\mathsf{Both}(b, a), \mathsf{Both}(b, c)), \mathsf{Either}(\mathsf{Both}(c, a), \mathsf{Both}(c, b)))$$

This can be seen by drawing a tree of possibilities.

On the other hand if we have a free choice followed by a forced choice then we get the expression

$$\mathsf{Either}(\mathsf{Maybe}(\mathsf{Both}(a, b), \mathsf{Both}(a, c)), \mathsf{Maybe}(\mathsf{Both}(b, a), \mathsf{Both}(b, c)), \mathsf{Maybe}(\mathsf{Both}(c, a), \mathsf{Both}(c, b)))$$

This begs the question of whether

$$\mathsf{Maybe}(\mathsf{Either}(X), \ldots, \mathsf{Either}(Y)) \stackrel{?}{=} \mathsf{Either}(\mathsf{Maybe}(X), \ldots, \mathsf{Maybe}(Y)).$$

In words, this is asking whether it matters if I can freely choose between uncertain outcomes or whether it is uncertain which free choice I will be able to make.

Observe that in any case

$$\mathsf{Either}(\mathsf{Both}(a, b), \mathsf{Both}(a, c)) = \mathsf{Both}(a, \mathsf{Either}(b, c))$$

$$\mathsf{Maybe}(\mathsf{Both}(a,b),\mathsf{Both}(a,c)) = \mathsf{Both}(a,\mathsf{Maybe}(b,c))$$

So we can reduce the first expression to

$$\mathsf{Maybe}(\mathsf{Both}(a,\mathsf{Either}(b,c)),\mathsf{Both}(b,\mathsf{Either}(a,c)),\mathsf{Both}(c,\mathsf{Either}(a,b)))$$

The second expression reduces to

$$\mathsf{Either}(\mathsf{Both}(a,\mathsf{Maybe}(b,c)),\mathsf{Both}(b,\mathsf{Maybe}(a,c)),\mathsf{Both}(c,\mathsf{Maybe}(a,b)))$$

In the example of choosing from $[a,b,c]$ it seems like the distinction is immaterial, as long as there is no dependency between a,b,c. In both cases I can guarantee only one resource. If there is some dependency, things may be more complicated. Suppose $\mathsf{Both}(a,b)$ is highly desirable, but $\mathsf{Both}(a,c)$ is catastrophic. All other outcomes are neutral. If you had a free choice first, you wouldn't want to choose a, because, then the catastrophic outcome might be realized by the forced choice. So in this case, you would always end up with a neutral outcome, if you want to avoid catastophe at all costs. On the other hand, if the forced choice was first, there is a chance you get a and are able to realize the desirable outcome. Moreover, you can always avoid catastrophe.

This could be visualized by two agents each attempting to satisfy a list of preferences by adding items to a common pool. Depending on what types of items appear on the list, it might make a difference in what order they proceed. However, it seems that a difference only manifests if they have compound items on their lists.

4.2.3 Substitution Rules

We want to extend the obtaining relation $a \to b$ to bundles. We do this via the following substitution rules:

- If $a \to x$ and you have $\mathsf{Both}(a,b)$, then you can get $\mathsf{Both}(x,b)$
- If $a \to x$ and you have $\mathsf{Either}(a,b)$, then you can get $\mathsf{Either}(x,b)$.
- If I have $\mathsf{Either}(a,b)$, I can obtain a

- If $a \to x$, and you have $\mathsf{Maybe}(a, b)$, then you can get $\mathsf{Maybe}(x, b)$.
- If I have a, I can obtain $\mathsf{Maybe}(a, b)$

Moreover, we have the following simplification rules:

- $\mathsf{Either}(a, a) = a = \mathsf{Maybe}(a, a)$

4.2.4 Debt

We can introduce a notion of debt by saying that for a resource a, $\neg a$ signifies that I owe an a. If $a \to b$, then $\neg b \to \neg a$. Assuming two parties, $\neg\neg a = a$. If Maybe is uncertain because it is the choice of my opponent, then $\neg\mathsf{Either}(a, b) = \mathsf{Maybe}(\neg a, \neg b)$. Equivalently, thinking in terms of unknowns, if I owe a $\mathsf{Maybe}(a, b)$, then I can choose to provide either a, or b. That is $\neg\mathsf{Maybe}(a, b) = \mathsf{Either}(\neg a, \neg b)$.

Finally, lets think about how debt interacts with Both. It seems that owing both a and b independently can be expressed as $\mathsf{Both}(\neg a, \neg b)$. On the other hand $\neg\mathsf{Both}(a, b)$ would mean owing independent a and b. It is hard to find a difference between these two formulations. Furthermore we should have $\mathsf{Both}(a, \neg a) \to [\ ]$ and $[\ ] \to \mathsf{Both}(a, \neg a)$.

4.2.5 Free Bundles

Let $\mathcal{P}$ be a countable preorder expressing obtainability. Form $\mathbf{Bundle}(\mathcal{P})$ by adding all finite expressions obtainable by $\neg$, Both, Either, and Maybe, along with all arrows obtainable from the substitution rules. We wish to include also the empty expression $[\ ]$, which acts as the unit for Both.

Proposition 4.1. $\mathbf{Bundle}(\mathcal{P})$ *is compact closed with all finite non-empty meets and joins.*

Proof. Multiplication is given by Both, which is an endofunctor because of the substitution rules. Moreover, we noted above that have $\mathsf{Both}(a, \neg a) \to [\ ]$ and $[\ ] \to \mathsf{Both}(a, \neg a)$. The yanking equations are trivially satisfied because we are dealing with a preorder.

Finite joins are given by Maybe:

$$a_1 \vee a_2 = \mathsf{Maybe}(a_1, a_2)$$

We have injections $a_i \to \mathsf{Maybe}(a_1, a_2)$. Suppose $a_i \to t$ for $i = 1, 2$. Assume I have $\mathsf{Maybe}(a_1, a_2)$ then by two substitutions I can get $\mathsf{Maybe}(t, t) = t$. Hence $\mathsf{Maybe}(a_1, a_2) \to t$.

In a similar manner, finite meets are given by Either. ⅂⅃⅂

Warning It is important that this compact closed preorder not have all finite meets and joins, as otherwise they would coincide by the paper of Robin Houston [Hou06]. For example, if you had top $\top$ and bottom $\bot$, then $\mathsf{Both}(\top, \bot) \cong \bot$, because having an additional copy of $\top$ doesn't make any difference if you have the universal resource $\bot$. On the other hand, by negation and symmetry, $\top = \neg\bot \cong \neg\mathsf{Both}(\top, \bot) = \mathsf{Both}(\neg\top, \neg\bot) = \mathsf{Both}(\top, \bot)$. Hence $\top = \bot$, which in particular has the consequence that $a \cong b$ for any a, b, meaning any a would be obtainable from any b.

4.2.6 Extending Feasibility Relations to Bundles

Suppose we are given preorders $\mathcal{R}$ and $\mathcal{F}$, with a feasibility relation $\Phi : \mathcal{R} \nrightarrow \mathcal{F}$. We may then extend Φ to a feasibility relation $\bar{\Phi} : \mathbf{Bundle}(\mathcal{R}) \nrightarrow \mathbf{Bundle}(\mathcal{F})$ as follows.

We view $\mathbf{Bundle}(X)$ as an iterated construction starting with $X_0 = X$. Define

$$\mathsf{Both}(Y) = \{\mathsf{Both}(x, y) \mid x, y, \in X\},$$
$$\mathsf{Maybe}(Y) = \{\mathsf{Maybe}(x, y) \mid x, y, \in X\},$$
$$\mathsf{Either}(Y) = \{\mathsf{Either}(x, y) \mid x, y, \in X\},$$

where in each case we enforce $\mathsf{Op}(a, b) = \mathsf{Op}(b, a)$ for $\mathsf{Op} \in \{\mathsf{Both}, \mathsf{Maybe}, \mathsf{Either}\}$.

Then set

$$X_{i+1} = X_i \cup \mathsf{Both}(X_i) \cup \mathsf{Maybe}(X_i) \cup \mathsf{Either}(X_i).$$

Finally,

$$\mathbf{Bundle}(X) = \bigcup_{i=0}^{\infty} X_i,$$

and add arrows according to the preorder generated by the substitution, introduction, and elimination rules.

We can now extend Φ in a stepwise fashion. Suppose we have extended Φ to a feasibility relation $\Phi_i : \mathcal{R}_i \nrightarrow \mathcal{F}_i$. Extend this to $\Phi_{i+1} : \mathcal{R}_{i+1} \nrightarrow \mathcal{F}_{i+1}$ as follows. For $(r, f) \in \mathcal{R}_i^{\mathrm{op}} \times \mathcal{F}_i$ set

$$\Phi_{i+1}(r, f) = \Phi_i(r, f)$$

$$\Phi_{i+1}(\neg r, \neg f) \Leftrightarrow \neg\Phi_i(r, f),$$

$$\Phi_{i+1}(\mathsf{Both}(r_1, r_2), \mathsf{Both}(f_1, f_2)) \Leftrightarrow (\Phi_i(r_1, f_1) \wedge \Phi_i(r_2, f_2)) \vee (\Phi(r_1, f_2) \wedge \Phi(r_2, f_1))$$

$$\Phi_{i+1}(\mathsf{Either}(r_1, r_2), \mathsf{Maybe}(f_1, f_2)) \Leftrightarrow \bigvee_{\substack{l,k \in \{1,2\} \\ l \neq k}} \Phi_i(r_l, f_k),$$

$$\Phi_{i+1}(\mathsf{Maybe}(r_1, r_2), \mathsf{Either}(f_1, f_2)) \Leftrightarrow \bigwedge_{\substack{l,k \in \{1,2\} \\ l \neq k}} \Phi_i(r_l, f_k).$$

We now need to show that this assignment makes Φ_{i+1} a feasibility relation, provided that Φ_i is a feasibility relation. By assumption, monotonicity is guaranteed on terms up to complexity i. Because the arrows in $\mathcal{R}_{i+1}, \mathcal{F}_{i+1}$ are generated by the substitution, elimination, and introduction rules, it is sufficient to check monotonicity for these elementary cases.

We check the extended assignments for monotonicity:

($\neg$) Suppose $\Phi_{i+1}(\neg r, \neg f)$ and $x \to \neg r$, $\neg f \to y$ elementary. Then $x = \neg s$ for $r \to s$ and $y = \neg g$ for $g \to f$. Since $\Phi_{i+1}(\neg r, \neg f)$, we know $\neg\Phi_i(r, f)$. This implies $\neg\Phi_i(s, g)$, since Φ_i monotone. Hence $\Phi_{i+1}(\neg s, \neg g)$, that is $\Phi_{i+1}(x, y)$.

(Both) Suppose $\Phi_{i+1}(\mathsf{Both}(r_1, r_2), \mathsf{Both}(f_1, f_2))$ and $x \to \mathsf{Both}(r_1, r_2)$, $\mathsf{Both}(f_1, f_2) \to y$ elementary. Then $x = \mathsf{Both}(s_1, s_2)$ and $y = \mathsf{Both}(g_1, g_2)$. Thus by elementarity $s_1 \to r_1$, $s_2 \to r_2$ and $f_1 \to g_1$, $f_2 \to g_2$. Our assumption $\Phi_{i+1}(\mathsf{Both}(r_1, r_2), \mathsf{Both}(f_1, f_2))$ implies $(\Phi_i(r_1, f_1) \wedge \Phi_i(r_2, f_2))$ or $(\Phi(r_1, f_2) \wedge \Phi(r_2, f_1))$. Hence by monotonicity we get $(\Phi_i(s_1, g_1) \wedge \Phi_i(s_2, g_2))$ or $(\Phi(s_1, g_2) \wedge \Phi(s_2, g_1))$. This means $\Phi_{i+1}(\mathsf{Both}(s_1, s_2), \mathsf{Both}(g_1, g_2))$, as desired.

(Maybe **right**) Suppose $\Phi_{i+1}(\mathsf{Either}(r_1, r_2), \mathsf{Maybe}(f_1, f_2))$ and $x \to \mathsf{Either}(r_1, r_2)$, $\mathsf{Maybe}(f_1, f_2) \to y$ elementary. Then $x = \mathsf{Either}(s_1, s_2)$ and $y = \mathsf{Maybe}(g_1, g_2)$ with $f_1 \to g_1$, $f_2 \to g_2$ and $s_1 \to r_1$,$s_2 \to r_2$. By assumption $\Phi_i(r_l, f_k)$ for some $l, k \in \{1, 2\}$ with $l \neq k$ so $\Phi_i(s_l, g_k)$. Hence $\Phi_{i+1}(\mathsf{Either}(s_1, s_2), \mathsf{Maybe}(g_1, g_2))$.

(Maybe left) Suppose $\Phi_{i+1}(\mathsf{Maybe}(r_1, r_2), \mathsf{Either}(f_1, f_2))$ and $x \to \mathsf{Maybe}(r_1, r_2)$, $\mathsf{Either}(f_1, f_2) \to y$ elementary. Then $x \in \{r_1, r_2\}$ and $y \in \{f_1, f_2\}$. By assumption $\Phi_i(r_l, f_k)$ for all $l, k \in \{1, 2\}$ with $l \neq k$. So in all cases $\Phi_{i+1}(r_l, f_k) = \Phi_i(r_l, f_k)$.

This shows that Φ_{i+1} is again a feasibility relation. Observe that in the proof we only needed the operations for defining (Both) and (Maybe right) to be monotone on **Bool**, while in the case of negation we used the fact that **Bool** has only two values, and in the case of (Maybe left) we needed a conjunction.

Now, by induction on i, we can extend a feasibility relation $\Phi : \mathcal{R} \twoheadrightarrow \mathcal{F}$ to $\Phi_\infty : \mathbf{Bundle}(\mathcal{R}) \twoheadrightarrow \mathbf{Bundle}(\mathcal{F})$, as we intended.

4.2.7 Relation to Queries

The construction above gives us formulae for queries. Let $\Phi : \mathcal{R} \twoheadrightarrow \mathcal{F}$. First fix a resources $r_1, r_2 \in \mathbf{Bundle}(\mathcal{R})$ and consider $h(r) = \{f \in \mathbf{Bundle}(\mathcal{F}) | \Phi_\infty(r, f)\}$.

If $f \in h(r_1) \cup h(r_2)$, then there are i, j such that $\Phi_i(r_1, f)$ or $\Phi_j(r_2, f)$. Assuming $i \leq j$ this means $\Phi_j(r_1, f)$ or $\Phi(r_2, f)$. Hence $\Phi_{j+1}(\mathsf{Either}(r_1, r_2), f)$, so $\Phi_\infty(\mathsf{Either}(r_1, r_2), f)$. Therefore $h(r_1) \cup h(r_2) \subseteq h(\mathsf{Either}(r_1, r_2))$. Conversely, if $f \in h(\mathsf{Either}(r_1, r_2))$ there is an i such that $\Phi_i(\mathsf{Either}(r_1, r_2), f)$. Then $\Phi_{i-1}(r_1, f)$ or $\Phi_{i-1}(r_2, f)$. Hence $f \in h(r_1) \cup h(r_2)$. In conclusion, $h(\mathsf{Either}(r_1, r_2)) = h(r_1) \cup h(r_2)$.

If $f \in h(r_1) \cap h(r_2)$, then there are i, j such that $\Phi_i(r_1, f)$ and $\Phi_j(r_2, f)$. Assuming $i \leq j$ this means $\Phi_j(r_1, f)$ and $\Phi(r_2, f)$. Hence $\Phi_{j+1}(\mathsf{Maybe}(r_1, r_2), f)$. This shows $h(r_1) \cap h(r_2) \subseteq h(\mathsf{Maybe}(r_1, r_2))$. Conversely, if $f \in h(\mathsf{Maybe}(r_1, r_2))$ there is i such that $\Phi_i(\mathsf{Maybe}(r_1, r_2), f)$. Hence $\Phi_{i-1}(r_1, f)$ and $\Phi_{i-1}(r_2, f)$, so $f \in h(r_1) \cap h(r_2)$. In conclusion, $h(\mathsf{Maybe}(r_1, r_2)) = h(r_1) \cap h(r_2)$.

Querying for fixed functionalities yields dual results: $h'(\mathsf{Maybe}(f_1, f_2)) = h'(f_1) \cup h'(f_2)$ and $h'(\mathsf{Either}(f_1, f_2)) = h'(f_1) \cap h'(f_2)$.

4.2.8 Induced Operations on Feasibility Relations

The structure on $\mathbf{Bundle}(\mathcal{R})$, $\mathbf{Bundle}(\mathcal{F})$ allows us to define operations on parallel feasibility relations $\Phi, \Psi : \mathbf{Bundle}(\mathcal{R}) \twoheadrightarrow \mathbf{Bundle}(\mathcal{F})$. Let $r_1, r_2 \in \mathbf{Bundle}(\mathcal{R})$ and $f_1, f_2 \in \mathbf{Bundle}(\mathcal{F})$

($\neg$) Set

$$(\neg\Phi)(r, f) \Leftrightarrow \neg\Phi(\neg r, \neg f).$$

If $s \to r$, $f \to g$ then $\neg r \to \neg s$ and $\neg g \to \neg f$. Hence $\neg\Phi(\neg r, \neg f)$ implies $\neg\Phi(\neg s, \neg g)$, so $(\neg\Phi)(s, g)$.

(Both) Set

$$\mathsf{Both}(\Phi, \Psi)(\mathsf{Both}(r_1, r_2), \mathsf{Both}(f_1, f_2)) \Leftrightarrow \bigvee_{\substack{i,j,k,l \in \{1,2\} \\ i \neq k, j \neq l}} (\Phi(r_i, f_j) \wedge \Psi(r_k, f_l))$$

(Maybe) Set

$$\mathsf{Maybe}(\Phi, \Psi)(\mathsf{Either}(r_1, r_2), \mathsf{Maybe}(f_1, f_2)) \Leftrightarrow$$

(Either) Set

$$\mathsf{Either}(\Phi, \Psi)(\mathsf{Maybe}(r_1, r_2), \mathsf{Either}(f_1, f_2)) \Leftrightarrow$$

4.3 A General Scheme to Extend Co-Design

The above suggests the following recipe for extending Co-Design:

- Introduce new resource/functionality objects derived from some original resources/functionalities.
- Select some subset of these that cohere with the obtainability structure that is already in place.
- Specify how the obtainability relation extends to derived objects
- Generate a derived resource/functionality preorder containing the derived objects together with the new relations
- Characterize this structure algebraically and check correspondence to logical structures
- Extend feasibility relations between original structures to the derived structures

4.4 Dependent Choice

4.5 Valued Resources

Here we will assign the things in our preorder P values in some commutative quantale $\mathcal{Q}$. Thus the derived objects are functions $v_p : \{p\} \to \mathcal{Q}$, for $p \in \mathcal{Q}$. Assigning these values in a coherent way with respect to obtainability means that if $p \to q$ then $v_p(p) \leq v_q(q)$. Hence a coherent assignment is just a monotone map $v : P \to \mathcal{Q}$.

Suppose we have fixed such a coherent valuation v. We can then introduce derived valuations on bundles. For each operation in $\mathcal{Q}$, we get a distinct type of bundle. Formally obtain these by the composition

$$P \times P \to \mathcal{Q} \times \mathcal{Q} \xrightarrow{\star} \mathcal{Q},$$

where $\star$ is some monotone operation on $\mathcal{Q}$. This means that between bundles of the same type we have a coherent substitution rule: If $p \to q$ and you have $\star(p, r)$, then you can get $\star(q, r)$. This could also be formulated $p \to q$ and $p' \to q'$ imply $\star(p, q) \to \star(p', q')$.

We now need to specify the remaining introduction / elimination rules. For this we name the operations in $(\mathcal{Q}, \otimes, \wedge, \vee)$. We set

- $\wedge(p, q) \to p$
- $r \to p$ and $r \to q$ imply $r \to \wedge(p, q)$
- $p \to \vee(p, q)$
- $p \to r$ and $q \to r$ imply $\vee(p, q) \to r$

These rules are coherent because $\wedge$ and $\vee$ are meet and join in $\mathcal{Q}$.

Now form all finite expressions using $\otimes$, $\wedge$, and $\vee$ and add all arrows induced by the substitution, introduction and elimination rules to yield $\mathbf{Val}(P)$.

Next show that we actually have a monotone map $\mathbf{Val}(P) \to \mathcal{Q}$ and that $\mathbf{Val}(P)$ has finite meets and joins.

References

[Hou06] Robin Houston. "Finite Products are Biproducts in a Compact Closed Category". In: *Journal of Pure and Applied Algebra* 212.2 (2006), pp. 394–400. DOI: 10.1016/j.jpaa.2007.05.021. arXiv: 0604542 [math]. URL: http://arxiv.org/abs/math/0604542http://dx.doi.org/10.1016/j.jpaa.2007.05.021.

[MZ18] Dan Marsden and Maaike Zwart. "Quantitative foundations for resource theories". In: *Leibniz International Proceedings in Informatics, LIPIcs* 119.32 (2018), pp. 1–17. ISSN: 18688969. DOI: 10.4230/LIPIcs.CSL.2018.32.

[SD04] Andrea Schalk and Valeria De Paiva. "Poset-valued sets or how to build models for linear logics". In: *Theoretical Computer Science* 315.1 (2004), pp. 83–107. ISSN: 03043975. DOI: 10.1016/j.tcs.2003.11.014.

[Yet90] David N. Yetter. "Quantales and (noncommutative) linear logic". In: *Journal of Symbolic Logic* 55.1 (1990), pp. 41–64. ISSN: 0022-4812. DOI: 10.2307/2274953.