Linear Logic and Co-Design Notebook

Marius Furter

May 25, 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

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

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.

$$\begin{array}{c} \dfrac{\Gamma \vdash B}{\Gamma, A \vdash B}\ (\text{weakening Left}) \\[2ex] \dfrac{\Gamma, A \vdash B}{\Gamma \vdash A \multimap B}\ (\multimap \text{ Right}) \end{array} \quad \rightsquigarrow \quad \dfrac{\dfrac{\Gamma \vdash B}{\Gamma, A \vdash B}\ (\text{weakening Left})}{\Gamma \vdash A \multimap B}\ (\multimap \text{ Right})$$

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

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

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

$$\frac{}{A \vdash A}\ (\mathrm{Axiom})$$

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

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

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

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

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

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

regard them as a derivation

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

and compose with f to get a derivation

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

which corresponds to a tuple of proofs of sequents

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

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

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

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

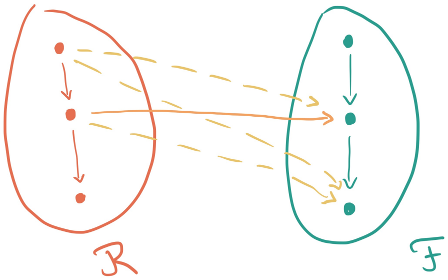
Let $\mathcal{R}$ and $\mathcal{F}$ be posets. $\mathcal{R}$ represents resources and $\mathcal{F}$ represents functionalities. For $a, b \in \mathcal{R}$, we interpret $a \leq b$ as meaning "if I have a, then I also have

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 $\mathcal{R}^{\text{op}} \times \mathcal{F} \to \mathbf{Bool}$, where **Bool** denotes the poset generated by `false` $\to$ `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

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

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.

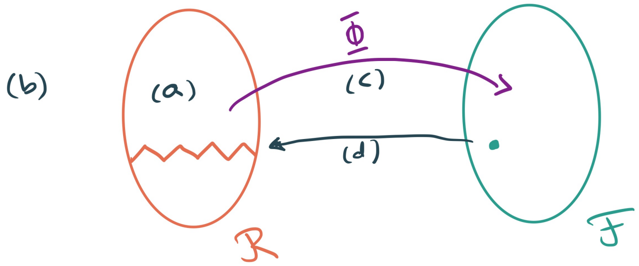
Lemma 2.1. *If all ascending chains in $\mathcal{R}$ are finite, then the map $h : \mathcal{F} \to \mathsf{A}\mathcal{R}$ is monotone. That is, if $f \to g$, then $\downarrow f_{max} \subseteq \downarrow g_{max}$.*

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 $\downarrow a_{\max} = \mathbb{Z}_{\leq 0} \nsubseteq \emptyset = \downarrow b_{\max}$.

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 $L_h = h'^{-1}(\texttt{true})$ is now a lower set: If $y \in L$ and $x \to y$, then $h'(x) \to h'(y) = \texttt{true}$, so $h'(x) = \texttt{true}$ since $\texttt{true}$ 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 $h(r) = \texttt{true}$, then also $g(r) = \texttt{true}$, 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 $l(x) = \texttt{true}$ 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 $l(x) = \texttt{false}$, then $l(y) = \texttt{false}$, because $x \notin L \Rightarrow y \notin L$ by the contrapositive of the lower set property. Similarly, if $l(y) = \texttt{true}$, then also $l(x) = \texttt{true}$. 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:

$$\begin{tikzcd}
\mathcal{R} \arrow[d, "\shortmid" marking] \arrow[dr, bend left] \arrow[dr, phantom, "\Swarrow"] & \\
\mathcal{F} \arrow[r] & \mathbf{Bool}
\end{tikzcd}$$

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 $\mathbf{Pos_{Bool}}$ 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 $\mathbf{Pos_{Bool}}$. 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 $\mathbf{Pos_{Bool}}$

2.5.1 Generalized Querying

We can generalize the notion of query within $\mathbf{Pos_{Bool}}$. 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 $\mathbf{Pos_{Bool}}$.

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 $\mathbf{Pos_{Bool}}$ 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 $\mathbf{Pos_{Bool}}$.

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 $\mathbf{Pos}_{\mathbf{Bool}}$.

$$\begin{tikzcd}
\mathcal{R} \arrow[d, "|" marking] \arrow[rrd, bend left] & & \\
\mathcal{F} \arrow[rr] \arrow[rru, phantom, "\Swarrow"] & & \mathbf{Bool}
\end{tikzcd}$$

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 $\mathbf{Pos}_{\mathbf{Bool}}$.

$$\begin{tikzcd}
\mathcal{R} \arrow[d, "|" marking] \arrow[rrd, "\rho" description] \arrow[rrd, bend left=50] \arrow[rrd, phantom, bend left=25, "\Swarrow"] & & \\
\mathcal{F} \arrow[rr] \arrow[rr, phantom, bend left=20, "\Downarrow"] & & \mathbf{Bool}
\end{tikzcd}$$

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 $\texttt{false}$, 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 $\texttt{false}$ 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 $\texttt{false}$. 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. $\square$

2.6.1 Describing Other Queries

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

$$\begin{tikzcd}
\mathcal{R} \arrow[d, "\shortmid" marking] \arrow[rrd] \arrow[rd, phantom, "\Downarrow"] & & \\
\mathcal{F} \arrow[rr, "\phi" description] \arrow[rr, bend right=50] \arrow[rr, phantom, bend right=25, "\Downarrow"] & & \mathbf{Bool}
\end{tikzcd}$$

The univesal φ would satisfy: For any φ' making Φ a morphism in $\mathbf{Pos}_{\mathbf{Bool}}$, 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 ($\mathcal{Q}$-enriched Functor)**.** If $\mathcal{A}$ and $\mathcal{B}$ are $\mathcal{Q}$-enriched categories, then a $\mathcal{Q}$*-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 $\mathcal{Q}$-enriched categories and whose morphisms are $\mathcal{Q}$-enriched functors.

Because $\mathcal{Q}$ is thin, there can be at most one natural transformation between any two $\mathcal{Q}$-functors.

Definition 3.4 ($\mathcal{Q}$-enriched Natural Transformation)**.** Given parallel $\mathcal{Q}$-enriched functors $F, G : \mathcal{A} \to \mathcal{B}$, the presence of a $\mathcal{Q}$-*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 $\mathcal{Q}$ 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 $\mathcal{Q}$ 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 $\mathbf{Cat}(\mathcal{Q})$.

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

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

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 $\Phi(a,b) = \mathtt{true}$, 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 $\varovee : \mathcal{M} \circledast \mathcal{M} \to \mathcal{M}$ and a unit $\mathcal{Q}$-functor $\mu : \mathcal{I} \to \mathcal{M}$ making the following diagrams commute:

$$\begin{tikzcd}
\mathcal{M} \circledast \mathcal{M} \circledast \mathcal{M} \arrow[r, "\mathcal{M}\circledast\varovee"] \arrow[d, "\varovee\circledast\mathcal{M}"] & \mathcal{M} \circledast \mathcal{M} \arrow[d, "\varovee"] \\
\mathcal{M} \circledast \mathcal{M} \arrow[r, "\varovee"] & \mathcal{M}
\end{tikzcd}
\qquad
\begin{tikzcd}
\mathcal{M} \circledast \mathcal{I} \arrow[r, "\mathcal{M}\circledast\mu"] \arrow[dr, "\sim"'] & \mathcal{M} \circledast \mathcal{M} & \mathcal{I} \circledast \mathcal{M} \arrow[l, "\mu\circledast\mathcal{M}"'] \arrow[dl, "\sim"] \\
& \mathcal{M} &
\end{tikzcd}$$

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

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

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

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

Proposition 3.2. *Let $(\mathcal{Q}, \circledast, k)$ be commutative quantale, and $(\mathcal{V}, \varovee, \mu)$ 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*

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

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

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,

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

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,

$\mathit{right} : E(A \circledast \mathcal{I}) \cong E(A)$, $\mathit{left} : E(\mathcal{I} \circledast A) \cong E(A)$ 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.

$\dashv$

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:

$$\begin{tikzcd}
EP \arrow[d, "\shortmid" marking] \arrow[rrd, bend left] & & \\
ER \arrow[rr] & {} \arrow[from=1-2, to=2-2, Rightarrow] & \mathcal{Q}
\end{tikzcd}$$

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 $\mathbf{Cat}(\Omega)$ and $\mathbf{Prof}(\Omega)$ 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 $\mathbf{Rel}_{\mathbf{Pos}}$:

$$\begin{tikzcd}
E(A \times B) \arrow[r, "\vartheta_{A,B}"] \arrow[d, "E(\Phi\times\Psi)"] & E(A) \times E(B) \arrow[d, "E\Phi\times E\Psi"] \\
E(A' \times B') \arrow[r, "\vartheta_{A',B'}"] & E(A') \times E(B')
\end{tikzcd}$$

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 Ω induces a functor $\bullet : (\mathbf{Prof}(\mathbb{B}) \wr_E \Omega)^{\times n} \to \mathbf{Prof}(\mathbb{B}) \wr_E \Omega$ as follows:*

Objects *$\alpha_i : EA_i \to \Omega$ are mapped to the composite*

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

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 $\bullet : \Omega^{\times n} \to \Omega$ be the monotone n-ary operation in question. Suppose $\alpha_i : EA_i \to \Omega$, $\beta_i : EB_i \to \Omega$ 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.

$$\begin{tikzcd}
E(A_1 \times \ldots \times A_n) \arrow[r, "\vartheta_{A_1,\ldots,A_n}"] \arrow[dd, "E(\Phi_1 \times \ldots \times \Phi_n)"] & E(A_1) \times \ldots \times E(A_n) \arrow[dd, "E\Phi_1 \times \ldots \times E\Phi_n"] \arrow[dr, "\alpha_1 \times \ldots \times \alpha_n"] & & \\
& & \mathcal{Q}^{\times n} \arrow[r, "\bullet"] & \mathcal{Q} \\
E(B_1 \times \ldots \times B_n) \arrow[r, "\vartheta_{B_1,\ldots,B_n}"] & E(B_1) \times \ldots \times E(B_n) \arrow[ur, "\beta_1 \times \ldots \times \beta_n"'] & &
\end{tikzcd}$$

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

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

$$left : \mathbb{1} \times A \to A,$$
$$right : A \times \mathbb{1} \to A,$$
$$swap : A \times B \to B \times A.$$

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:

$$\begin{tikzcd}
E((A\times B)\times C) \arrow[r, "\vartheta_{A,B}\vartheta_{(A\times B),C}"] \arrow[d, "E(assoc)"] & (E(A)\times E(B))\times E(C) \arrow[r, "(\alpha\times\beta)\times\gamma"] \arrow[d, "assoc"] & (\mathcal{Q}\times\mathcal{Q})\times\mathcal{Q} \arrow[d, "assoc"] \arrow[dr, "\otimes\text{ twice}"] & \\
E(A\times(B\times C)) \arrow[r, "\vartheta_{B,C}\vartheta_{A,(B\times C)}"'] & E(A)\times(E(B)\times E(C)) \arrow[r, "\alpha\times(\beta\times\gamma)"'] & \mathcal{Q}\times(\mathcal{Q}\times\mathcal{Q}) \arrow[r, "\otimes\text{ twice}"'] & \mathcal{Q}
\end{tikzcd}$$

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 $\mathcal{Q}$ in **Pos**.

The diagram for the left unitor is:

$$\begin{tikzcd}
E(\mathbb{1}\times A) \arrow[r, "\vartheta_0\vartheta_{\mathbb{1},A}"] \arrow[dr, "E(left)"'] & \mathbb{1}\times E(A) \arrow[r, "\mathbb{1}\times\alpha"] \arrow[d, "left"] & \mathbb{1}\times\mathcal{Q} \arrow[r, "k\times\mathcal{Q}"] \arrow[d, "left"] & \mathcal{Q}\times\mathcal{Q} \arrow[dl, "\otimes"] \\
 & E(A) \arrow[r, "\alpha"] & \mathcal{Q} &
\end{tikzcd}$$

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 $\mathcal{Q}$. The diagram for the right unitor is analogous.

Finally, for commutativity we have

$$\begin{tikzcd}
E(A\times B) \arrow[r] \arrow[d, "E(swap)"] & E(A)\times E(B) \arrow[r] \arrow[d, "swap"] & \mathcal{Q}\times\mathcal{Q} \arrow[d, "swap"] \arrow[dr, "\otimes"] & \\
E(B\times A) \arrow[r] & E(B)\times E(A) \arrow[r] & \mathcal{Q}\times\mathcal{Q} \arrow[r, "\otimes"] & \mathcal{Q}
\end{tikzcd}$$

where the left square commutes because E is symmetric oplax monoidal, the middle by naturality of *swap*, and the right triangle by commutativity of $\mathcal{Q}$. $\dashv$

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

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

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

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

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

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

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

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

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:

$(\Phi_{i+1}(r, f) \Leftrightarrow \Phi_i(r, f))$ Suppose $\Phi_{i+1}(r, f)$ and $x \to r$, $f \to y$ elementary. If x or y have complexity i, we are covered by the induction hypothesis. Otherwise $x = \mathsf{Either}(r, s)$ and $y = \mathsf{Maybe}(f, g)$. By the fourth rule we have $\Phi_{i+1}(\mathsf{Either}(r, s), \mathsf{Maybe}(f, g))$, as desired.

$(\neg)$ 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 operation for defining (Both) 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. We needed the (Maybe left) rule to be a disjunction, in order to get monotonicity for the elimination rules.

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

4.2.7 Functoriality

4.2.8 Relation to Queries

The construction above gives us formulae for queries. Let $\Phi : \mathcal{R} \nrightarrow \mathcal{F}$. 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:

$$h'(\mathsf{Maybe}(f_1, f_2)) = h'(f_1) \cup h'(f_2) \text{ and } h'(\mathsf{Either}(f_1, f_2)) = h'(f_1) \cap h'(f_2).$$

Let's now turn to Both, fixing resources. If $f \in h(\mathsf{Both}(r_1, r_2))$, then $f = \mathsf{Both}(f_1, f_2)$ and there is i such that $(\Phi_i(r_1, f_1) \wedge \Phi_i(r_2, f_2)) \vee (\Phi_i(r_1, f_2) \wedge \Phi_i(r_2, f_1))$. Thus $f_1 \in h(r_1)$ and $f_2 \in h(r_2)$, or $f_1 \in h(r_2)$ and $f_2 \in h(r_1)$. Hence $f = \mathsf{Both}(f_1, f_2) \in \{\mathsf{Both}(x, y) | (x \in h(r_1) \wedge y \in h(r_2)) \vee (x \in h(r_2) \wedge y \in h(r_1))\}$. Conversely, if f is in the set just described, then $f \in h(\mathsf{Both}(r_1, r_2))$. Therefore,

$$h(\mathsf{Both}(r_1, r_2)) = \{\mathsf{Both}(x, y) | (x \in h(r_1) \wedge y \in h(r_2)) \vee (x \in h(r_2) \wedge y \in h(r_1))\}.$$

By symmetry, the same holds for fixed functionalities.

Finally, fixing resources, $f \in h(\neg r)$ iff $\neg\Phi(r, \neg f)$ holds. Thus

$$h(\neg r) = \{f \mid \neg f \notin h(r)\}.$$

4.2.9 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 $\Phi, \Psi : \mathbf{Bundle}(\mathcal{R}) \nrightarrow \mathbf{Bundle}(\mathcal{F})$. 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

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

Suppose $\mathsf{Both}(\Phi, \Psi)(\mathsf{Both}(r_1, r_2), \mathsf{Both}(f_1, f_2))$ and $x \to \mathsf{Both}(r_1, r_2)$, $\mathsf{Both}(f_1, f_2) \to y$ elementary. If $x = \mathsf{Both}(s_1, s_2)$ and $y = \mathsf{Both}(g_1, g_2)$ with $s_i \to r_i$ and $f_i \to g_i$. By the monotonicity of all the operators involved in the defining expression we have $\mathsf{Both}(\Phi, \Psi)(\mathsf{Both}(s_1, s_2), \mathsf{Both}(g_1, g_2))$. On the other hand, if $x = \mathsf{Either}(\mathsf{Both}(r_1, r_2), s)$ and $y = \mathsf{Maybe}(\mathsf{Both}(f_1, f_2), g)$, then Does not work!

(Maybe) Set

$$\mathsf{Maybe}(\Phi, \Psi)(r, f) \Leftrightarrow \Phi(r, f) \wedge \Psi(r, f)$$

(Either) Set

$$\mathsf{Either}(\Phi, \Psi)(r, f) \Leftrightarrow \Phi(r, f) \vee \Psi(r, f)$$

Both of these are monotone because they are a composition

$$R^{\mathrm{op}} \times F \xrightarrow{\Delta} R^{\mathrm{op}} \times R^{\mathrm{op}} \times F \times F \cong R^{\mathrm{op}} \times F \times R^{\mathrm{op}} \times F \xrightarrow{\Phi\times\Psi} \mathbf{Bool} \times \mathbf{Bool} \xrightarrow{\star} \mathbf{Bool}.$$

Using the monotonicity and the arrows in the Bundle we can get statements such as

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

On the other hand if $\mathsf{Maybe}(\Phi, \Psi)(\mathsf{Either}(r_1, r_2), \mathsf{Maybe}(f_1, f_2))$, then $\Phi(\mathsf{Either}(r_1, r_2), \mathsf{Maybe}(f_1, f_2))$ and $\Psi(\mathsf{Either}(r_1, r_2), \mathsf{Maybe}(f_1, f_2))$. If Φ and Ψ are induced, then this means that $\bigvee_{l,k\in\{1,2\}} \Phi_i(r_l, f_k)$ and $\bigvee_{l,k\in\{1,2\}} \Psi_i(r_l, f_k)$. This is equivalent to $\bigvee_{i,j,k,l\in\{1,2\}} \Phi(r_i, f_j) \wedge \Psi(r_k, f_l)$.

Some other definitions, see how these compare.

(Maybe) Set

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

(Either) Set

$$\mathsf{Either}(\Phi, \Psi)(\mathsf{Maybe}(r_1, r_2), \mathsf{Either}(f_1, f_2)) \Leftrightarrow \bigvee_{X_l\in\{\Phi,\Psi\}} \bigwedge_{i,j\in\{1,2\}} X_l(r_i, f_j)$$

4.2.10 Queries on the Induced Operations

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.

5 Guarantees

We find that free and forced choice is discussed in the computer science literature under the names “angelic” and “demonic” choice. The following draws from [MCR07] and [Mor04].

The previous section on bundles showed the difficulty with attempting to introduce choice at a term level. In particular, the extension of feasibility relations to this new structure presented problems. The following approach avoids this issue by working with upper and lower set functors. The key idea will be to move from feasibility relations between preorders of resources and functionalities to feasibility relations between lower sets. This is analogous to the move from relations to multirelations (relations on $A \times \mathsf{P}(B)$).

5.1 Fixing Interpretation

Since the following construction is quite flexible with respect to its interpretation it will be helpful to fix a rigid interpretation that we will stick to during theory development. Alternatives in interpretation will be highlighted when they arise.

5.1.1 Resources, Functionalities, and Processes

As always, we consider preordered sets of resources and functionalities as our basic entities. The preordering will be interpreted as expressing “free transformations” that are available to all agents involved. Specifically, $a \to b$ means that given a, I can get b instantly and for free. Alternatively, we can also interpret an arrow $a \to b$ as meaning that having a implies having b.

We note that what are considered resources and functionalities depends on what process we are considering. For example, if we have a process $\Phi : \mathcal{R} \rightsquigarrow \mathcal{F}$, we think of the resources $\mathcal{R}$ as being something we need to provide to Φ to obtain the functionalities $\mathcal{F}$.

We will think of Φ as being operated by an external agent. In practice this could be a company offering some service. On the other hand, we will think of ourselves as being in control of what resources we provide Φ and which specific service we choose. However, we may not know in advance which resources we will have available at the time we need to use Φ. Our control stems from the fact that we can choose amongst the resources available to us.

Observe that even if we control a process ourselves, there may be limitations as to its exact outcome. So the assumption that the process is in control of an external agent isn't necessary. However, it will help keep separate what is in our control and what is not.

5.2 Certain Processes

Under the interpretation above we can view traditional feasibility relations as those processes possessing certain outcomes. Let $\Phi : \mathcal{R} \nrightarrow \mathcal{F}$ be a feasibility relation. We think of Φ as specifying a contract between the operator of Φ and ourselves. The values Φ takes specify the transformations that are contractually available. Whenever $\Phi(r, f)$ is true, the service provider guarantees that if we deliver them an r, they will provide us with an f. The monotonicity condition just expresses that the service provider exploits all the transformations that are induced by the free transformations in $\mathcal{R}$ and $\mathcal{F}$: If they can guarantee an f given an r, they should also be able to guarantee an f given a resource that freely transforms into r. Similarly, if given r they can guarantee an f, and f freely transforms into g, they should also be able to guarantee $\Phi(r, g)$.

5.2.1 Composition for Certain Processes

We now consider a sequence of processes $\mathcal{A} \overset{\Phi}{\nrightarrow} \mathcal{B} \overset{\Psi}{\nrightarrow} \mathcal{C}$. For clarity we think of Φ and Ψ as being operated by distinct agents. Suppose we are in possession of some $a \in \mathcal{A}$. We want to think about our options for transforming a into some $c \in \mathcal{C}$. This is possible iff there is some $b \in \mathcal{B}$ such that $\Phi(a, b)$ and $\Psi(b, c)$. In this case the operator of Φ guarantees that b given our a, and the operator of Ψ guarantees a c given the intermediary b. We could go as far as imagining a third agent as offering us the composite and then outsourcing the individual tasks to Φ and Ψ respectively.

We can also think about composing processes in parallel: Given $\Phi, \Psi : \mathcal{R} \nrightarrow \mathcal{F}$, we can form $\Phi \cup \Psi : \mathcal{R} \nrightarrow \mathcal{F}$ and $\Phi \cap \Psi : \mathcal{R} \nrightarrow \mathcal{F}$. The former combines all the guarantees of Φ and Ψ, while the latter only takes those that are shared by between them. We may interpret $\Phi \cup \Psi$ as being able to access both the services of Φ and Ψ simultaneously. $\Phi \cap \Psi$ can be interpreted as not knowing in advance which provider will be available to us, so we can only guarantee the transformations that both provide.

5.3 Uncertain Processes

We will now generalize the above considerations to the case where the requirements for and outcomes of a transformation are no longer certain. In this case, a provider Φ offers a more general type of contract: Given a resource $r \in A \subseteq \mathcal{R}$, Φ guarantees to provide a functionality $f \in B \subseteq \mathcal{F}$. In other words, Φ is willing to accept more than just a specific resource for a transformation, but also only guarantees that the outcome will be amongst some set.

We loose nothing by assuming that the sets in question are be lower sets in $\mathcal{R}$ and $\mathcal{F}$. If Φ can guarantee $f \in B$ given some $r \in A$, and moreover s freely transforms into r, then they should also be able to guarantee $f \in B$ given s by first transforming s into r. Hence from both the point of view of the consumer and the provider, offering $\Phi(A, B)$ is equivalent to $\Phi(A \cup \{s\}, B)$. Conversely, suppose $f \in B$ and $g \to f$. If Φ were to produce g rather than a promised $h \in B$, they could freely transform g into $f \in B$ to satisfy the contract. So again the offer of $\Phi(A, B)$ is equivalent to $\Phi(A, B \cup \{g\})$ from both perspectives.

The possible options for transformations offered by Φ assemble into a relation $\mathsf{L}(\mathcal{R}) \overset{\Phi}{\nrightarrow} \mathsf{L}(\mathcal{F})$, where $(\mathsf{L}(P), \subseteq)$ is the preorder of lower sets ordered by inclusion. In fact, this is a feasibility relation: If Φ can guarantee $f \in B$ given $r \in A$ and $B \subseteq B'$, then certainly Φ should also be able to guarantee $f \in B'$ given $r \in A$. Similarly if $A' \subseteq A$, then given $r \in A'$, Φ should be able to guarantee $r \in B$.

5.3.1 Relation to Certain Processes

Observe that using the embeddings $\mathcal{R} \to \mathsf{L}(\mathcal{R})$ which send $r \mapsto \downarrow r$, we can obtain a traditional feasibility relation by composing $\mathcal{R}^{op} \times \mathcal{F} \to \mathsf{L}(\mathcal{R})^{op} \times \mathsf{L}(\mathcal{F}) \to \mathbf{Bool}$. This describes exactly those transformations available in Φ which are certain.

Conversely, given a traditional feasibility relation $\Phi : \mathcal{R} \nrightarrow \mathcal{F}$, we can lift it to a feasibility relation $\hat{\Phi} : \mathsf{L}(\mathcal{R}) \nrightarrow \mathsf{L}(\mathcal{F})$ by setting

$$\hat{\Phi}(A, B) = \mathtt{true} \Leftrightarrow \forall a \in A \, \exists b \in B : \Phi(a, b).$$

Monotonicity follows immediately from the choice of quantifiers.

5.3.2 Composition for Uncertain Processes

Again consider a sequence of processes $\mathsf{L}(\mathcal{A}) \overset{\Phi}{\nrightarrow} \mathsf{L}(\mathcal{B}) \overset{\Psi}{\nrightarrow} \mathsf{L}(\mathcal{C})$. An external provider can guarantee that $f \in C \subseteq \mathcal{C}$ provided $r \in A \subseteq \mathcal{A}$ iff there is a $B \subseteq B$ such that $\Phi(A, B)$ and $\Psi(B, C)$. To implement the contract they would accept some $r \in A$ and use Φ to guarantee an outcome in B. They can then use Ψ to guarantee some $f \in C$.

We are again able to compose processes in parallel: Given $\Phi, \Psi : \mathcal{R} \nrightarrow \mathcal{F}$, we can form $\Phi \cup \Psi : \mathcal{R} \nrightarrow \mathcal{F}$ and $\Phi \cap \Psi : \mathcal{R} \nrightarrow \mathcal{F}$. The interpretation remains the same as above.

5.3.3 Operations on $\mathsf{L}(-)$

For any $\mathcal{R}$, the preorder $\mathsf{L}(\mathcal{R})$ comes equipped with the operations of union and intersection. Consider $\Phi : \mathsf{L}(\mathcal{R}) \nrightarrow \mathsf{L}(\mathcal{F})$. The provider could say that they accept either $r \in A$ or $r \in A'$ (our choice), and guarantee $f \in B$. This means we can provide any $r \in A \cup A'$. They could also say that they will accept $r \in A$ or $r' \in A'$ but they will specify which. In this case we must be prepared to provide $r \in A \cap A'$. On the output side, the provider could say that they guarantee $f \in B$ or $f \in B'$, and they choose which. This is the same as them guaranteeing $f \in B \cup B'$. On the other hand, if we get to choose whether $f \in B$ or $f \in B'$, the provider must be prepared to provide $f \in B \cap B'$.

We will think of the provider Φ controlling $\mathsf{L}(\mathcal{F})$ and us as controlling $\mathsf{L}(\mathcal{R})$. Denote the free choice of the controller by Either and choice be an external agent by Maybe. Then $\mathsf{Either}(A, A') = A \cup A'$ and $\mathsf{Maybe}(A, A') = B \cap B'$.

There are some requirements that a provider should be able fulfil. If $\Phi(A, B)$ and $\Phi(A', B)$, then also $\Phi(A \cup A', B)$ since the provider doesn't know in advance which transformation we will choose to use. In this case also $\Phi(A \cap A', B)$ by monotonicity. On the other hand, if $\Phi(A, B)$ and $\Phi(A, B')$, then $\Phi(A, B \cap B')$, again because the provider doesn't know in advance which transformation we will pick. Again by monotonicity $\Phi(A, B \cup B')$. This shows that Φ is closed under Either and Maybe in the sense that

$$\Phi(A, B) \text{ and } \Phi(A', B) \Rightarrow \Phi(\mathsf{Maybe}(A, A'), B)$$

and similar for the other connectives and positions. The converse implications don't always hold. For example, from $\Phi(A \cap A', B)$, we can't necessarily deduce that $\Phi(A, B)$.

5.3.4 An Optimization Procedure

The results from the above section provide a way to optimize over feasibility relations. For each $B \in \mathsf{L}(\mathcal{F})$ consider $L_B := \{A : \Phi(A, B)\}$. By our reasoning above, we should also have $\Phi(\cup L_B, B)$ and this deal is in fact preferable to us as the consumer. In a similar fashion, for a fixed $A \in \mathsf{L}(\mathcal{R}$, set $L_A := \{B : \Phi(A, B)\}$ and deduce that $\Phi(A, \cap L_A)$. In particular, $\Phi(\cup L_B, \cap L_{\cup L_B})$ and $\Phi(\cup L_{\cap L_A}, \cap L_A)$ provide the optimal deals for a desired lower set of functionalities or available set of resources, respectively.

5.4 Uncertainty for Choice of Transformation

So far we have been considering a process $\Phi : \mathsf{L}(\mathcal{R}) \nrightarrow \mathsf{L}(\mathcal{F})$ and assuming that we can freely choose amongst the transformations Φ provides. It may however be the case that the provider can only guarantee such choice up to a set. Such a contract would look as follows: If you guarantee the provider that your choice of transformations will start at $L_R \in A \subseteq \mathsf{L}(\mathcal{R})$, they guarantee that you will be able to freely choose a guaranteed set amongst $L_F \in B \subseteq \mathsf{L}(\mathcal{F})$. If the provider can make such an offer, they should equivalently be able to offer $\Phi(\cup L_R, \cap L_F)$, since they don't know in advance which choice we will make. It therefore seems that such uncertainty is already captured in the framework we are considering. Formally, such higher order guarantees would be expressed as a feasibility relation $\Psi : \mathsf{L}(\mathsf{L}(\mathcal{R})) \nrightarrow \mathsf{L}(\mathsf{L}(\mathcal{F}))$, by similar considerations as we saw before. As we have argued, if $\Psi(L_R, L_F)$ is feasible, where L_R, L_F are a lower sets of lower sets, then the provider should also offer $\Psi(\cup L_R, \cap L_F)$. Hence any feasibility between sets can be reduced to that between singletons which is the same as a feasibility relation $\Phi : \mathsf{L}(\mathcal{R}) \nrightarrow \mathsf{L}(\mathcal{F})$.

5.5 Lifting to a Functor

We lift $\Phi : \mathsf{L}(\mathcal{R}) \nrightarrow \mathsf{L}(\mathcal{F})$ to a functor $\hat{\Phi} : \mathsf{U}(\mathsf{L}(\mathcal{R})) \to \mathsf{U}(\mathsf{L}(\mathcal{F}))$ by sending

$$U_R \mapsto \{l_F : \exists l_R \in U_R \, \Phi(l_R, l_F)\}.$$

The result is indeed an upper set since if $l_F \subseteq l'_F$, then for any $l_R \in U_R$, $\Phi(l_R, l_F)$ implies $\Phi(l_R, l'_F)$ by monotonicity. Moreover, $\hat{\Phi}$ is a functor because if $U_R \supseteq U'_R$ and $l_F \in \hat{\Phi}(U'_R)$, then the same l_R also witnesses that $l_F \in \hat{\Phi}(U_R)$.

We interpret the induced functor as follows: If we can guarantee that our resources lie in one of the $l_R \in U_R$, then we can guarantee that we can obtain a functionality in any of the $l_F \in \hat{\Phi}(U_R)$.

Note that $\hat{\Phi}$ is just an extension of the curried $\check{\Phi} : \mathsf{L}(\mathcal{R}) \to \mathsf{U}(\mathsf{L}(\mathcal{F}))$ to upper sets in the domain via an existential quantifier. We could also work directly with the $\check{\Phi}$, but the quantifier would then come into play when we try to compose.

Also the operation of mapping $\Phi \mapsto \hat{\Phi}$ is itself a functor since for $\mathsf{L}(\mathcal{R}) \overset{\Phi}{\nrightarrow} \mathsf{L}(\mathcal{F}) \overset{\Psi}{\nrightarrow} \mathsf{L}(\mathcal{G})$ we have

$$\begin{aligned}
(\widehat{\Psi\Phi})(U_R) &= \{l_G : \exists l_R \in U_R \; \Psi\Phi(l_R, l_G)\} \\
&= \{l_G : \exists l_R \in U_R \; \exists l_F : \Psi(l_R, l_F) \wedge \Phi(l_F, l_G)\} \\
&= \{l_G : \exists l_F : (\exists l_R \in U_R \; \Psi(l_R, l_F)) \wedge \Phi(l_F, l_G)\} \\
&= \hat{\Psi}\{l_F : \exists l_R \in U_R : \Phi(l_R, l_f)\} \\
&= \hat{\Psi}\hat{\Phi}
\end{aligned}$$

and $\hat{\mathrm{id}}(U_F) = \{l_F : \exists l'_F \in U_F : \mathrm{id}(l'_F, l_F)\} = \{l_F : \exists l'_F \in U_F : l'_F \subseteq l_F\} = U_F$, because U_F is an upper set.

There should also be a contravariant functor that is induced, which could be useful.

5.5.1 Partial Equivalence of Formulations

We have seen that any feasibility relation $\Phi : \mathsf{L}(\mathcal{R}) \nrightarrow \mathsf{L}(\mathcal{F})$ lifts to a functor $\hat{\Phi} : \mathsf{U}(\mathsf{L}(\mathcal{R})) \to \mathsf{U}(\mathsf{L}(\mathcal{F}))$. Conversely, given some $\Theta : \mathsf{U}(\mathsf{L}(\mathcal{R})) \to \mathsf{U}(\mathsf{L}(\mathcal{F}))$ we first get a functor $\mathsf{L}(\mathcal{R}) \to \mathsf{U}(\mathsf{L}(\mathcal{F}))$ by composing with the inclusion $\mathsf{L}(\mathcal{R}) \to \mathsf{U}(\mathsf{L}(\mathcal{R}))$, and then uncurry the result to get a profunctor $\check{\Theta} : \mathsf{L}(\mathcal{R}) \nrightarrow \mathsf{L}(\mathcal{F})$.

If we start with $\Phi : \mathsf{L}(\mathcal{R}) \nrightarrow \mathsf{L}(\mathcal{F})$, lift to $\hat{\Phi} : \mathsf{U}(\mathsf{L}(\mathcal{R})) \to \mathsf{U}(\mathsf{L}(\mathcal{F}))$, and compose $\mathsf{L}(\mathcal{R}) \to \mathsf{U}(\mathsf{L}(\mathcal{R})) \to \mathsf{U}(\mathsf{L}(\mathcal{F}))$, we map $l_R \mapsto \{l_F : \exists l'_R \in \uparrow l_R : \Phi(l'_R, l_F)\} = \{l_F : \Phi(l_R, l_F)\}$, by monotonicity. This is just the curried version of Φ, so we recover the original feasibility relation by uncurrying.

However, not every functor $\mathsf{U}(\mathsf{L}(\mathcal{R})) \to \mathsf{U}(\mathsf{L}(\mathcal{F}))$ is induced by some feasibility relation. For example, any functor mapping the empty upper set to a non-empty value cannot be induced.

5.5.2 Interpretation on the Upper Set Level

Fix $\Theta : \mathsf{U}(\mathsf{L}(\mathcal{R})) \to \mathsf{U}(\mathsf{L}(\mathcal{F}))$. Then $\Theta(U_R) = U_F$ expresses that if we can provide some r (our choice) in any of the $l_R \in U_R$ (not out choice which), then we can guarantee some f (not our choice) in the $l_F \in U_F$ of our choice.

Formulated as a transaction between us an a provider Θ this becomes the following: For each upper set U_R, the provider has a single deal. If you choose to accept the deal corresponding to U_R, the provider lets you choose amongst any $l_F \in \Theta(U_R)$. He then informs you of the $l_R \in U_R$ for which you must provide him with an $r \in l_R$ of your choice. The provider then gives you some $f \in l_F$ according to his discretion.

If your resources are obtained from some previous transaction Ω, then your choice of deal $U_R = \Omega(X)$ is forced by whatever X is.

Notice that it make a difference what U_R looks like internally. On one extreme, if $U_R = \{l_R\}$ is a singleton, you know you will just have to provide any $r \in l_R$. But if $U_R = \{\{r_i\}\}$ is a union of singletons, then you would have to be able to provide any r_i, even if $\cup\{r_i\} = l_R$.

Observe also that there is a general theme going on regarding free and forced choices. Whenever you are allowed to choose freely in the codomain, the corresponding choice is forced in the domain and vice versa. This is necessary for composition to make sense.

Along these lines we can also think of the transaction as a competition with ourselves. Suppose that we choose U_A to start with and choose $l_F \in \Theta(U_A)$. If we now want to do another transformation we are stuck with $l_F \in \Theta(U_A)$ by our previous decision, so that choice is forced. Now we can still choose a free $l_G \in \Omega\Theta(U_A)$, but this again forces our hand for the next step.

5.5.3 Operations on the Upper Set Level

Fix a map $\Theta : \mathsf{U}(\mathsf{L}(\mathcal{R})) \to \mathsf{U}(\mathsf{L}(\mathcal{F}))$ and upper set $U_R \in \mathsf{U}(\mathsf{L}(\mathcal{R}))$. If the provider offers you a choice between some of the $l_R \in U_R$, this corresponds to the union of the l_R. In formulae: $U_A = \{\mathsf{Either}\{l_i\}, l_j, \ldots\} = \{\cup\{l_i\}, l_j, \ldots\}$. Being distinct elements of U_R already expresses that the provider has the choice. So $\{\mathsf{Maybe}\{l_i\}, l_j, \ldots\} = \{l_i\} \cup \{l_j, \ldots\}$.

If we would offer up an additional choices U'_R to the provider, his options would then be $U_R \cup U'_R$.

If he offers us the choice between U_R and U'_R, he would have a free choice between $\{l_R \cup l'_R : l_R \in U_R, l'_R \in U'_R\}$. This is because both of us would be happy if I allowed the provider to chose any $\mathsf{Either}(l_R, l'_R)$: If I can provide a resource for each $l_R \in U_R$, or alternatively, for each $l'_R \in U'_R$, then I am also guaranteed to satisfy any $l_R \cup l'_R$ with my resources. Conversely, if the provider can guarantee us a free choice of some l_F no matter whether he gets the choice between the set of l_R or the set of l'_R, then there is a specific l_R and l'_R that will allow him to guarantee a specific l_F. Hence the union will also allow him to guarantee that l_F.

On the target side the operations are the same with the roles reversed.

In the source, we can think of an upper set U_A as being a big Maybe over all its elements l_R, which are in turn Eithers over their elements r_i. The laws above then follow if we grant that Maybe and Either distribute over one another.

5.6 Antichain Model

The upper-lower set model contains quite a lot of redundancy. On the lower set level, if there are both $a \to b$ in a given lower set, then a is redundant. On the source side this is because the provider needs to be able to produce a guaranteed functionality even if he is given b, and we as the consumer can always convert our a to a b. On the target side, if the provider guarantees a set containing a and b, then we must expect the worst, that is b. If we instead got an a, we could always convert it to our worst case expectation. Hence, we can consider worst case estimates rather than lower sets, which also contain the more optimistic options. This amounts to looking at antichains of resources and functionalities, because any for any pair $a \to b$ in our set, one of the options is redundant.

On the upper set level we have a similar form of redundancy. Whenever we have $l_R \subseteq l'_R$ in U_R, we can safely ignore l'_R. On the source side this is due to fact that we must be prepared to deliver $r \in l_R$, which automatically means we can provide l'_R. On the target side the provider must be prepared to provide an $f \in l_F$, which entails that he can also provide a $f \in l'_F \supseteq l_F$. Hence we are again led to consider antichains.

5.6.1 Profunctor Level

The antichain analogue of the lower set level considers profunctors $\Phi : \mathsf{A}(R) \nrightarrow \mathsf{A}(F)$, where the antichains are ordered as follows: $A \to A'$ iff $\downarrow A \subseteq \downarrow A'$. If I can guarantee $r \in A$ (or better) then $r \in \downarrow A$. If $\downarrow A' \subseteq \downarrow A'$, then guaranteeing $r \in A$ (or better) implies guaranteeing $r \in A'$ (or better).

The interpretation is almost the same: $\Phi(A, B)$ expresses that given a resource of our choice $r \in A$ (or better), the provider guarantees to produce a $f \in B$ (or better) of his choice. The (or better) clauses may be ignored if we are thinking in terms of guarantees. We must just be careful not to draw the conclusion that if we are guaranteed $f \in B$, that this means the provider will directly produce an f. The set that will satisfy the contract is actually $\downarrow B$ Conversely, if we have guarantee $r \in A$, the set of viable resources is $\downarrow A$.

5.6.2 Functor Level

As before we lift $\Phi : \mathsf{A}(R) \nrightarrow \mathsf{A}(F)$ to a functor $\hat{\Phi} : \mathsf{A}(\mathsf{A}(R)) \to \mathsf{A}(\mathsf{A}(F))$ where the anitchains are ordered by $A \to A'$ iff $\uparrow A \subseteq \uparrow A'$. We assign

$$A \mapsto \min\{a_F : \exists a_R \in A : \Phi(a_R, a_F)\},$$

This function is monotone if for all A in $\mathsf{A}(\mathsf{A}(R))$ we have

$$\uparrow \min\{a_F : \exists a_R \in A : \Phi(a_R, a_F)\} = \{a_F : \exists a_R \in A : \Phi(a_R, a_F)\}.$$

Note that $\{a_F : \exists a_R \in A : \Phi(a_R, a_F)\} = \{a_F : \exists a_R \in \uparrow A : \Phi(a_R, a_F)\}$ by monotonicity of Φ. So $\uparrow A \subseteq \uparrow A'$ implies $\{a_F : \exists a_R \in A : \Phi(a_R, a_F)\} \subseteq \{a_F : \exists a_R \in A' : \Phi(a_R, a_F)\}$. By the hypothesis this means $\hat{\Phi}(A) \to \hat{\Phi}(A')$.

5.6.3 Joins and Meets on the Profunctor Level

For the next part we assume that R satisfies the ascending chain condition.

Given two antichains $A, B \in \mathsf{A}(R)$ we can form $\max(A \cup B)$. We have $\downarrow A \subseteq \downarrow A \cup \downarrow B = \downarrow (A \cup B) = \downarrow \max(A \cup B)$, so $A \to \max(A \cup B)$ and similarly $B \to \max(A \cup B)$. Suppose that an antichain T satisfies $A \to T$ and $B \to T$. Then $\downarrow A \subseteq \downarrow T$ and $\downarrow B \subseteq \downarrow T$, whence $\downarrow A \cup \downarrow B \subseteq \downarrow T$. Since $\downarrow A \cup \downarrow B = \downarrow \max(A \cup B)$, we have shown $\max(A \cup B) \to T$. Therefore $\max(A \cup B)$ is the join of A and B.

Now consider $\max(\downarrow A \cap \downarrow B)$. We have $\downarrow A \subseteq \downarrow A \cap \downarrow B = \downarrow (\downarrow A \cap \downarrow B) = \downarrow \max(\downarrow A \cap \downarrow B)$, so $A \to \max(\downarrow A \cap \downarrow B)$. Similarly $B \to \max(\downarrow A \cap \downarrow B)$. Suppose $T \to A$ and $T \to B$. Then $\downarrow T \subseteq \downarrow A \cap \downarrow B = \downarrow \max(\downarrow A \cap \downarrow B)$, whence $T \to \max(\downarrow A \cap \downarrow B)$, showing that $\max(\downarrow A \cap \downarrow B)$ is the meet of A and B.

5.6.4 Joins and Meets on the Functor Level

For this part we must assume that R satisfies the ascending chain condition and $\mathsf{A}(R)$ satisfies the descending chain condition.

Given two antichains $A, B \in \mathsf{A}(\mathsf{A}(R))$ we can form $\min(A \cup B)$. In this case $A \to \min(A \cup B)$ since $\uparrow A \subseteq \uparrow A \cup \uparrow B = \uparrow (A \cup B) = \uparrow \min(\uparrow (A \cup B)) = \uparrow \min(A \cup B)$, and similarly $B \to \min(A \cup B)$. If $A \to T$ and $B \to T$, then $\uparrow A \cup \uparrow B \subseteq T$. Since $\uparrow A \cup \uparrow B = \uparrow \min(A \cup B)$, we see that $\uparrow \min(A \cup B)$ is the join of A and B.

Consider $\min(\uparrow A \cap \uparrow B)$. We have $\uparrow A \cap \uparrow B = \uparrow (\uparrow A \cap \uparrow B) = \uparrow \min(\uparrow A \cap \uparrow B)$. Using the fact that the intersection is the meet in the ordering of the induced upper sets, this shows that $\min(\uparrow A \cap \uparrow B)$ is the meet of A and B.

$\min(\{\max(a \cup b) : a \in A, b \in B\}) \overset{?}{=} \min(\uparrow A \cap \uparrow B)$

5.6.5 Distributivity

For $A, B, C \in \mathsf{A}(\mathsf{A}(R))$ we have

$$\begin{aligned}
A \wedge (B \vee C) &= \min(\uparrow A \cap \uparrow (\min(B \cup C))) \\
&= \min(\uparrow A \cap \uparrow (B \cup C)) \\
&= \min(\uparrow A \cap (\uparrow B \cup \uparrow C)) \\
&= \min((\uparrow A \cap \uparrow B) \cup (\uparrow A \cap \uparrow C))) \\
&= \min(\uparrow (\uparrow A \cap \uparrow B) \cup \uparrow (\uparrow A \cap \uparrow C))) \\
&= \min(\uparrow \min(\uparrow A \cap \uparrow B) \cup \uparrow \min(\uparrow A \cap \uparrow C))) \\
&= \min(\uparrow (\min(\uparrow A \cap \uparrow B) \cup \min(\uparrow A \cap \uparrow C))) \\
&= \min(\min(\uparrow A \cap \uparrow B) \cup \min(\uparrow A \cap \uparrow C)) \\
&= (A \wedge B) \vee (A \wedge C).
\end{aligned}$$

and

$$\begin{aligned}
A \vee (B \wedge C) &= \min(A \cup \min(\uparrow B \cap \uparrow C)) \\
&= \min(\uparrow (A \cup \min(\uparrow B \cap \uparrow C))) \\
&= \min(\uparrow A \cup \uparrow \min(\uparrow B \cap \uparrow C)) \\
&= \min(\uparrow A \cup \uparrow (\uparrow B \cap \uparrow C)) \\
&= \min(\uparrow A \cup (\uparrow B \cap \uparrow C)) \\
&= \min((\uparrow A \cup \uparrow B) \cap (\uparrow A \cup \uparrow C)) \\
&= \min(\uparrow (A \cup B) \cap \uparrow (A \cup C)) \\
&= \min(\uparrow \min(A \cup B) \cap \uparrow \min(A \cup C)) \\
&= (A \vee B) \wedge (A \vee C).
\end{aligned}$$

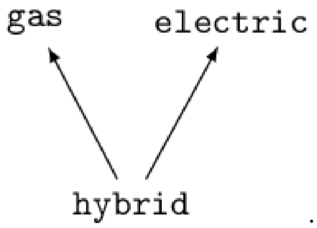
Therefore the meet and join operations distribute over one another. The analogous result should also hold for the profunctor level.

6 Extended Examples

6.1 Experimental Hybrid Motor

Situation A team is responsible for implementing the motor of a autonomous vehicle. There are two established options available: an electric motor offering good torque but limited range, and a gasoline motor offering poor torque but good range. The team further proposes an experimental hybrid design combining the strengths of both the established options. As of yet they are unsure whether they will be able to deliver on their promise.

Model We represent the options for the motors as the preorder R



The functionalities the motors can deliver is given as the preorder F

$$\begin{tikzcd}
& (\texttt{bad torque}, \texttt{bad range}) & \\
(\texttt{good torque}, \texttt{bad range}) \arrow[ur] & & (\texttt{bad torque}, \texttt{good range}) \arrow[ul] \\
& (\texttt{good torque}, \texttt{good range}) \arrow[ul] \arrow[ur] & .
\end{tikzcd}$$

The feasibility relation $\Phi : R \nrightarrow F$ between these preorders is generated by

$$\Phi(\texttt{hybrid}, (\texttt{good torque}, \texttt{good range})),$$

$$\Phi(\texttt{gas}, (\texttt{bad torque}, \texttt{good range})),$$

$$\Phi(\texttt{electric}, (\texttt{good torque}, \texttt{bad range})).$$

We want to model the fact that it is uncertain whether the team can provide the hybrid motor. To this end we consider collections of resources. For example, to express that the team can promise to deliver the gas motor or the hybrid motor, but they don't know which, we would use the collection $\{\texttt{gas}, \texttt{hybrid}\}$. Such collections may as well be lower sets since the team can always include any better resources in its promise, because it just has to deliver one of the indicated resources in the end. If the team does end up delivering one of the better resources, feasibility is preserved.

In the finite setting we can represent lower sets by antichains by taking maximal elements. Thus the collection $\{\texttt{gas}, \texttt{hybrid}\}$ becomes an antichain $(\texttt{gas})$ which generates the original lower set by taking its lower closure. We interpret this as the promise of the team to deliver $\texttt{gas}$ or better.

We can list all the possible promises the team can make and order them by $A \to B$ iff $\downarrow A \subseteq \downarrow B$:

$$\begin{tikzcd}
 & (\mathtt{gas}, \mathtt{electric}) & \\
(\mathtt{electric}) \arrow[ur] & & (\mathtt{gas}) \arrow[ul] \\
 & (\mathtt{hybrid}) \arrow[ul] \arrow[ur] & \\
 & () \arrow[u] & .
\end{tikzcd}$$

This ordering reflects both the values of the resources and the specificity of the promise. For example, a promise of (`electric` or better) is superior to the promise of (`gas` or `electric` or better) because the former is more specific. Note that a promise of an element in the empty set is promising the impossible, and hence is better than any other promise.

We can do the same transformation to the functionalities

$$\begin{tikzcd}
 & ((\mathtt{bad\ torque}, \mathtt{bad\ range})) & \\
 & ((\mathtt{bad\ torque}, \mathtt{good\ range}), (\mathtt{good\ torque}, \mathtt{bad\ range})) \arrow[u] & \\
((\mathtt{bad\ torque}, \mathtt{good\ range})) \arrow[ur] & & ((\mathtt{good\ torque}, \mathtt{bad\ range})) \arrow[ul] \\
 & ((\mathtt{good\ torque}, \mathtt{good\ range})) \arrow[ul] \arrow[ur] & \\
 & () \arrow[u] & .
\end{tikzcd}$$

The uncertainty here is inherited from that in the choice of motor. If the motor team doesn't know if they can implement the hybrid design, then the other teams also don't know whether the combination of $(\texttt{good torque}, \texttt{good range})$ will be possible.

We can now express feasibility on this higher level as follows. We want a pair of collections to be feasible iff the promise of a resource in the source implies the promise of a functionality in the target. For this to work there has to be a functionality obtainable in the target, regardless of which resource in the source is provided. Formally we set

$$\hat{\Phi}(A, B) \iff \forall r \in\downarrow A\, \exists f \in\downarrow B : \Phi(r, f).$$

This is again a feasibility relation since 'for all' is closed under taking subsets and 'exists' under taking supersets. Observe that by monotonicity of Φ we in fact have

$$\hat{\Phi}(A, B) \iff \forall r \in\downarrow A\, \exists f \in\downarrow B : \Phi(r, f) \iff \forall r \in A\, \exists f \in B : \Phi(r, f).$$

Instead of extending an existing feasibility relation in this way, we could also directly define feasibility on the collections. For example, for the induced relation we have

$$\hat{\Phi}((\texttt{gas}), (\texttt{bad torque}, \texttt{good range})),$$

$$\hat{\Phi}((\texttt{electic}), (\texttt{good torque}, \texttt{bad range})),$$

$$\hat{\Phi}((\texttt{gas}, \texttt{electic}), (\texttt{bad torque}, \texttt{good range}), (\texttt{good torque}, \texttt{bad range})).$$

However,if we directly declare feasibility we could also set

$$\Omega((\texttt{gas}), (\texttt{bad torque}, \texttt{good range})),$$

$$\Omega((\texttt{electic}), (\texttt{good torque}, \texttt{bad range})),$$

$$\Omega((\texttt{gas}, \texttt{electic}), (\texttt{bad torque}, \texttt{bad range})).$$

In particular,

$$\Omega((\texttt{gas}, \texttt{electic}), (\texttt{bad torque}, \texttt{good range}), (\texttt{good torque}, \texttt{bad range})) = \texttt{false}.$$

This choice could model some dependency that is not reducible to feasibility at the lower level. Maybe, if it is unknown whether gas or electric will be provided it is not possible to guarantee either ($\mathtt{bad\ torque}, \mathtt{good\ range}$) or ($\mathtt{good\ torque}, \mathtt{bad\ range}$) because a chassis that could fit both motors inevitably gives ($\mathtt{bad\ torque}, \mathtt{bad\ range}$).

The relation Ω cannot be induced since we know that its values on the singletons gas and electric are the same as those of $\hat{\Phi}$. If Ω were induced by ω, then ω must agree with Φ on the singletons. But in that case $\Omega = \hat{\omega} = \hat{\Phi}$, a contradiction. This shows that we can capture effects by declaring feasibility on this higher level that are not reducible to feasibility on the lower level.

Next we consider the choice structure on the source:

$$\begin{tikzcd}
& (\mathtt{gas}, \mathtt{electric}) & \\
(\mathtt{electric}) \arrow[ur] & & (\mathtt{gas}) \arrow[ul] \\
& (\mathtt{hybrid}) \arrow[ul] \arrow[ur] & \\
& () \arrow[u] &
\end{tikzcd} .$$

The motor team can combine their promises. Let A and B represent two promises. The team can promise (A or B), that is they promise that the resource will lie in A or better, or in B or better, but they are not specifying which. This is they same as promising a resource in $\downarrow A \cup \downarrow B$, which is again a lower set. The antichain that corresponds to this is $\max(\downarrow A \cup \downarrow B)$, which is the same as $\max(A \cup B)$ since $\downarrow A \cup \downarrow B = \downarrow (A \cup B)$ and $\max(\downarrow X) = \max(X)$.

For example, if the team promises electric or better, or gas or better, this is the same as promising ($\mathtt{gas}, \mathtt{electric}$) or better.

On the other hand, the motor team could combine promises in a different way. They could stipulate that we as managers get to chose between which promise (A or B) they must fulfill. Representing this formally requires another level of choice: We as managers promise to chose among as set of promises. For example, we say that we will choose among $\{(\mathtt{electric}), (\mathtt{gas}, \mathtt{electric})\}$. These sets may as well be upper sets, since adding more options is good for us as managers, and at the same time adding less specific promises doesn't constrain the team either. Again we can represent the upper sets as antichains by taking minima. We order this collection by $A \to B$ iff $\uparrow A \supseteq \uparrow B$:

$$\begin{tikzcd}
 & () & \\
 & ((\mathtt{gas}, \mathtt{electric})) \arrow[u] & \\
((\mathtt{electric})) \arrow[ur] & & ((\mathtt{gas})) \arrow[ul] \\
 & ((\mathtt{electric}), (\mathtt{gas})) \arrow[ul] \arrow[ur] & \\
 & ((\mathtt{hybrid})) \arrow[u] & \\
 & (()) \arrow[u] & .
\end{tikzcd}$$

This order takes into account the specificity of our promise as well as the values of the team's promises they contain.

By the interpretation we've set up so far, the team giving us as managers the choice between which promise they need to fulfill corresponds to unioning the represented upper sets: If A, B are team promises then $(A), (B)$ are the managerial promises that correspond to them. If the team gives us as managers the choice between $(A), (B)$, this corresponds to $((A), (B))$. In general, managerial choice between existing managerial choices M, N is given by $\min(\uparrow M \cup \uparrow N) = \min(M \cup N)$.

We've now expressed managerial choice, but what about team choice at this level. Let A, B be managerial promises. We give the team the choice between which managerial promise we will draw from. Denote managerial choice over a set X by $\mathsf{M}(X)$ and denote team choice over a set Y by $\mathsf{T}(Y)$. In our formal framework we have a standard representation of choice as $\mathsf{M}(\mathsf{T}Y_1, \ldots, \mathsf{T}Y_n)$. We observe that the two types of choice should distribute over on another (the reasoning behind this will be explained elsewhere). Hence, for sets Y_i, Z_i of resources we have

$$\begin{aligned}\mathsf{T}(\mathsf{M}(\mathsf{T}Y_1, \ldots, \mathsf{T}Y_n), \mathsf{M}(\mathsf{T}Z_1, \ldots, \mathsf{T}Z_m)) &= \mathsf{M}(\{\mathsf{T}(\mathsf{T}(Y_i), \mathsf{T}(Z_j)) : 1 \leq i \leq n, 1 \leq j \leq m\}) \\ &= \mathsf{M}(\{\mathsf{T}(Y_i \cup Z_j) : 1 \leq i \leq n, 1 \leq j \leq m\})\end{aligned}$$

Therefore, we can write a team choice between managerial promises A, B as $\min\{\max(a \cup b) : a \in A, b \in B\}$. If A, B are singletons, this reduces to the usual team choice between two team promises.

Returning to the preorder

$$\begin{tikzcd}
& () & \\
& ((\texttt{gas}, \texttt{electric})) \arrow[u] & \\
((\texttt{electric})) \arrow[ur] & & ((\texttt{gas})) \arrow[ul] \\
& ((\texttt{electric}), (\texttt{gas})) \arrow[ul] \arrow[ur] & \\
& ((\texttt{hybrid})) \arrow[u] & \\
& (()) \arrow[u] & .
\end{tikzcd}$$

we observe that the meets of the elements are exactly managerial choice, while the joins are team choice.

We may again lift a profunctor $\Omega : \mathsf{A}(R) \to \mathsf{A}(F)$ from the previous level to this level as follows: A pair (A, B) is feasible iff for every promise of functionalities $b \in B$, there is a promise of resources $a \in A$ such that $\Omega(a, b)$. Hence (A, B) being feasible expresses that if we as managers demand a $b \in B$, there is $a \in A$ that we can select for which the teams can implement b based on a. Formally

$$\tilde{\Omega}(A, B) \iff \forall b \in\uparrow B\, \exists a \in\uparrow A : \Omega(a, b) \iff \forall b \in B\, \exists a \in A : \Omega(a, b),$$

where the last equivalence follows by monotonicity of Ω. The fact that 'exists' is preserved by supersets and 'for all' by subsets implies that $\tilde{\Omega}$ is a profunctor.

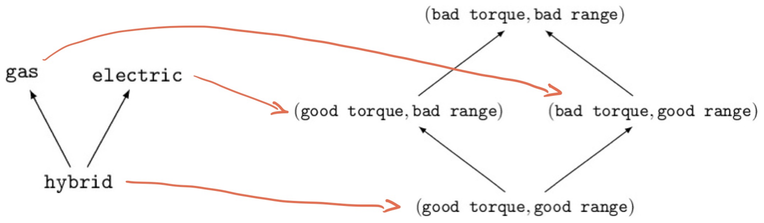
Instead of inducing a profunctor from the lower team level, we could also directly set a profunctor at this upper level. The interpretation for setting (A, B) feasible is as follows: If we as managers haven't yet decide whether to go with a specific $b \in B$, the collection of options A is sufficient to guarantee that we can realize any choice we end up making. The manager is looking to keep his options open in the target and hence dependency is traveling from target to source. In the case of the teams, they want to keep options open in the source, so dependency travels in the opposite direction.

Let's see what types of questions we can ask at the managerial level. Let $\Psi : \mathsf{A}(\mathsf{A}(R)) \nrightarrow \mathsf{A}(\mathsf{A}(F))$ be a profunctor at the managerial level. To start with lets consider $\Phi : R \nrightarrow F$ generated by

$$\Phi(\texttt{hybrid}, (\texttt{good torque}, \texttt{good range})),$$

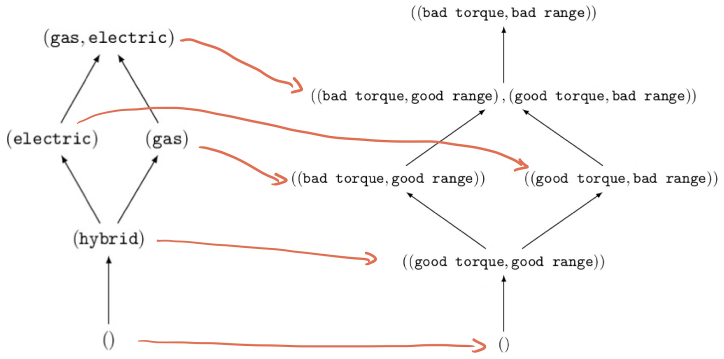
$$\Phi(\texttt{gas}, (\texttt{bad torque}, \texttt{good range})),$$

$$\Phi(\texttt{electric}, (\texttt{good torque}, \texttt{bad range})).$$



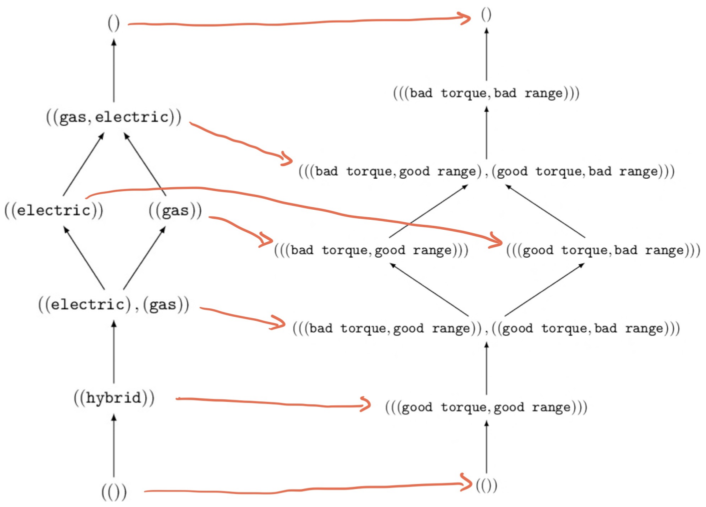
.

The induced feasibility $\hat{\Phi} : \mathsf{A}(R) \to \mathsf{A}(F)$ on the team level is


((bad torque, bad range))
(gas, electric)
((bad torque, good range), (good torque, bad range))
(electric)
(gas)
((bad torque, good range))
((good torque, bad range))
(hybrid)
((good torque, good range))
()
()

.

The induced feasibility relation $\tilde{\Phi} : \mathsf{A}(\mathsf{A}(R)) \to \mathsf{A}(\mathsf{A}(F))$ on the managerial level is


()
()
(((bad torque, bad range)))
((gas, electric))
(((bad torque, good range), (good torque, bad range)))
((electric))
((gas))
(((bad torque, good range)))
(((good torque, bad range)))
((electric), (gas))
(((bad torque, good range)), ((good torque, bad range)))
((hybrid))
(((good torque, good range)))
(())
(())

At the managerial level we can see that if we as managers want to choose between either (`bad torque`, `good range`) or (`good torque`, `bad range`), we can do so if we can force our team to produce either `electric` or better, or `gas` or better. We could also do so if we were able to force our team to produce `hybrid`.

On the other hand, if we leave the choice of (`bad torque`, `good range`) or (`good torque`, `bad range`) to the teams, (corresponds to (((`bad torque`, `good range`), (`good torque` then it is sufficient for the team to be able to produce (`gas`, `electric`) or better.

In our example in extending Φ we observe that the generators of Φ get extended along meets and joins to produce generators of the $\tilde{\hat{\Phi}}$. A such there is an optimality present in the induced feasibility relation not present in an arbitrary feasibility relation at the managerial level.

6.2 Experimental Hybrid Motor 2

Situation The team is still uncertain whether they can produce the hybrid motor. In addition, they don't know if the motor will deliver (`goodtorque`, `goodrange`) or something weaker.

Model We express the uncertainty in the results that the hybrid motor may provide by modifying the feasibility relation. For this we observe that the set of feasibility relations $R \nrightarrow F$ is preorded by containment. This ordering expresses that if $\Phi \to \Psi$, then Φ can freely be transformed into Ψ in the sense that if (a, b) is feasible in Ψ, the same transformation can be achieved using Φ. Therefore, we can put the choice structure presented above on the set of feasibility relations.

To see how this structure can be related to the choice structure on the source and target, we attempt to take it into account when extending feasibility relations to the team and managerial levels.

Suppose $L = \{\Psi_i\}$ is a family of feasibility relations $R \nrightarrow F$. The team promises they will be able to implement one of the Ψ_i, but not which. Again, we lose nothing by assuming that L is a lower set, since if the team can always make additional promises they don't intend to fulfill, and if they do deliver a 'better' (larger) feasibility relation, they can do still deliver on what they originally promised (and more).

We now lift the collection L to the team level. The uncertainty in which Ψ_i will be chosen will be reflected in the uncertainty present in the promises of functionality the team can make. We modify the lifting from above: A pair A, B of lower sets is feasible iff for every $a \in A$ and every $\Psi_i \in L$ there is $b \in B$ such that $\Psi_i(a, b)$. That is regardless of which resource in the collection ends up being provided and regardless of which transformation is implemented, we can produce some functionality in B. Formally, and using antichains X, Y representing the lower sets,

$$\begin{aligned} \hat{L}(X, Y) &\iff \forall x \in \downarrow X \, \forall \Psi_i \in L \, \exists \downarrow y \in Y : \Psi_i(x, y) \\ &\iff \forall x \in X \, \forall \Psi_i \in L \, \exists y \in Y : \Psi_i(x, y). \end{aligned}$$

For example, if $\Psi_1(\texttt{hybrid}, (\texttt{good torque}, \texttt{good range}))$ and $\Psi_2(\texttt{hybrid}, (\texttt{good torque}, \texttt{bad rang}$
then $\hat{L}((\texttt{hybrid}), ((\texttt{good torque}, \texttt{good range}), (\texttt{good torque}, \texttt{bad range})))$.

Hence at the team level, uncertainty in choice of feasibility creates uncertainty in the target. If uncertainty is already present in the source, it is inherited and expanded.

Next we look at how to lift a collection of feasibility relations to the managerial level. Here we assume we have a collection $U = \{\Theta_i\}$ of profunctors at the team level. This collection represents managerial choice, that is, we as managers will decide which specific Θ_i will be used. We may as well assume these are upper sets, since we can always include options we don't intend to choose, and the team will still be able to implement any weaker choice, if we do end up making it. Recall that being a manager is about keeping options in the target open. Thus a pair A, B of upper sets representing managerial choices should be feasible, if regardless of which $b \in B$ we end up deciding on, there is $\Theta_i \in U$ and $a \in A$ such that $\Theta_i(a, b)$. Formally, in terms of antichains X, Y representing upper sets,

$$\begin{aligned} \tilde{U}(X, Y) &\iff \forall y \in \uparrow Y \exists \Theta_i \in U \exists x \in \uparrow X : \Theta_i(x, y) \\ &\iff \forall y \in Y \exists \Theta_i \in U \exists x \in X : \Theta_i(x, y). \end{aligned}$$

Because we have more options to choose from, we need less choices in the source to enable the same choices in the target.

We have thus shown that if we are given a collection

$$U = \{L : L \text{ a lower set of profunctors } R \nrightarrow F\},$$

we can flatten it to $\hat{U} = \{\hat{L} : \hat{L}\mathsf{A}(R) \nrightarrow \mathsf{A}F\}$ and finally to a single profunctor $\tilde{U} : \mathsf{A}(\mathsf{A}(R)) \nrightarrow \mathsf{A}(\mathsf{A}F)$. Hence the choice structure on collections of profunctors $R \nrightarrow F$ translates to operations on induced profunctors $\mathsf{A}(\mathsf{A}(R)) \nrightarrow \mathsf{A}(\mathsf{A}F)$.

Our next goal will be to flatten collections at the team (resp. managerial) level to a single profunctor at the same level. This flattening should restrict to the above on singletons. This will allow us to think about choices between porfunctors at those levels. For simplicity we switch back to just thinking about lower and upper sets, rather than the antichains representing them.

Let $P = \{\Phi_i\}$ be a collection of profunctors $\mathsf{L}(R) \nrightarrow \mathsf{L}(F)$ at the team level. We want to express team choice between the $\{\Phi_i\}$. Call the flattened profunctor Φ. We want a pair A, B of team choices to be feasible with respect to Φ iff for all $a \in A$ there is $\Phi_i \in P$, $X \in \mathsf{L}(R)$, and $Y \subseteq B$ such that $a \in X$, $\Phi_i(X, Y)$. That is no matter what $a \in A$ gets delivered, there is a Φ_i that can transform that a into an element guaranteed to lie in a subset of B. Formally,

$$\Phi(A, B) \iff \forall a \in A \exists \Phi_i \in P \exists a \in X \in \mathsf{L}(R) \exists Y \subseteq B \in \mathsf{L}(F) : \Phi_i(X, Y).$$

Since the Φ_i are feasibility relations $\Phi_i(X, Y)$ and $Y \subseteq B$ implies $\Phi_i(X, B)$. Moreover, $a \in X$ is equivalent to $\downarrow a \subseteq X$. Hence if $a \in X$ and $\Phi_i(X, B)$, then $\Phi_i(\downarrow a, B)$. So the above definition is in fact equivalent to the much simpler

$$\Phi(A,B) \iff \forall a \in A \exists \Phi_i \in P : \Phi_i(\downarrow a, B).$$

Φ is a profunctor because 'for all' is preserved by restriction to subsets and if $B \sup B'$ and $\Phi_i(\downarrow a, B)$ then also $\Phi_i(\downarrow a, B')$ by profunctoriality of Φ_i.

We now do the same thing at the managerial level. Let $P = \{\Phi_i\}$ be a collection of profunctors $\mathsf{U}(\mathsf{L}(R)) \nrightarrow \mathsf{U}(\mathsf{L}(F))$ at the managerial level. These represent choices we get to make as managers. We flatten these to a single profunctor $\Phi : \mathsf{U}(\mathsf{L}(R)) \nrightarrow \mathsf{U}(\mathsf{L}(F))$ as follows. A pair A, B should be feasible iff for every $b \in B$ there is Φ_i, $b \in Y \in \mathsf{U}(\mathsf{L}(F))$, and $X \subseteq A$ such that $\Phi_i(X, Y)$. That is, for any choice we make in the target, there is a profunctor Φ_i that allows us to find $x \in X \subseteq A$, that transforms into b via Φ_i. Formally,

$$\Phi(A,B) \iff \forall b \in B \exists \Phi_i \in P \exists b \in Y \in \mathsf{U}(\mathsf{L}(F)) \exists X \subseteq A : \Phi_i(X,Y).$$

As above, we note that $b \in Y \iff \uparrow b \subseteq Y$. Thus if $\Phi_i(X, Y)$, then also $\Phi_i(X, \uparrow b)$ since Φ_i is a profunctor and the upper sets are ordered by containment. Moreover, if $X \subseteq A$ and $\Phi_i(X, Y)$, then $\Phi_i(A, Y)$. Hence the above definition is equivalent to

$$\Phi(A,B) \iff \forall b \in B \exists \Phi_i \in P : \Phi_i(A, \uparrow b).$$

We have managed to reduce team choice at the team level and managerial choice at the managerial level. It remains to see how team choice looks at the managerial level.

Let $C\{\{\Phi_i\}_j\}$ be a collection of profunctors representing a managerial choice of team choices. Following the above definitions it seems we should set

$$\Phi(A,B) \iff \forall b \in B \exists \{\Phi_i\}_j \in C : \forall \Phi_i \in \{\Phi_i\}_j \exists b \in Y \exists A \subseteq X : \Phi_i(X,Y)$$
$$\iff \forall b \in B \exists \{\Phi_i\}_j \in C : \forall \Phi_i \in \{\Phi_i\}_j : \Phi_i(A, \uparrow b).$$

We should now check that given a choice structure on profunctors $R \nrightarrow F$, it is the same if we first lift them individually to the managerial level and then flatten, or perform the flattening lifts to each subsequent level.

On second thought, the condition should probably reach into every level and be of the form

$$\Phi(A,B) \iff \forall b \in B \exists a \in A \exists \{\Phi_i\}_j : \forall r \in a \exists \Phi_i \in \{\Phi_i\}_j \exists f \in b : \Phi_i(\uparrow\downarrow r, \uparrow\downarrow f).$$

This is saying that for every choice b we can make in the target as managers there is a choice of feasibilities $\{\Phi_i\}_j$ and a choice a in the source such that the the pair (a,b) is implementable by the team using some $\Phi_i \in \{\Phi_i\}_j$ of their choice. That is, regardless which resource $r \in a$ the team has available, there is a functionality $f \in b$ an implementation Φ_i allowing f to be obtained from r.

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.

[MCR07] C. E. Martin, S. A. Curtis, and I. Rewitzky. "Modelling angelic and demonic nondeterminism with multirelations". In: *Science of Computer Programming* 65.2 (2007), pp. 140–158. ISSN: 01676423. DOI: 10.1016/j.scico.2006.01.007.

[Mor04] Joseph M. Morris. "Augmenting types with unbounded demonic and angelic nondeterminacy". In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 3125 (2004), pp. 274–288. ISSN: 16113349. DOI: 10.1007/978-3-540-27764-4_15. URL: https://link.springer.com/chapter/10.1007/978-3-540-27764-4_15.

[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.