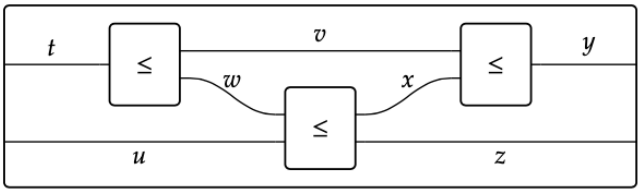
Chapter 2

Monoidal preorders and Enrichments

- Idea: discuss how to express recipes
- Bool category
- Cost category
- Between parallel lines apply the given product or sum function
- Continuous wire is equivalent to =
- To define a symmetric monodical preorder 4 things are required
  - Underlying set
  - An order imposed on it
  - A function
  - An identity element
- If (a to e) are satisfied than it is a monodical preorder

Semantics of wiring diagrams

- Example of a wiring diagram, the boxes represent some order or some process and the wires are you input/output elements



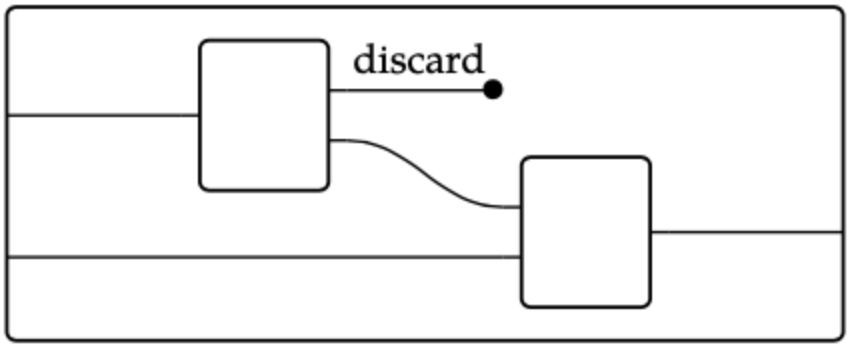
(2.15)

The inner boxes in Eq. (2.15) translate into the assertions:

$$t \leq v + w \qquad w + u \leq x + z \qquad v + x \leq y \tag{2.16}$$

- Some real-life examples

- Chemistry: similar to manufacturing but the discard axiom doesn’t hold here
- Manufacturing: discard axiom x<I You can trash anything you want it disappears from view



discard represented in a wiring diagram

- Informatics: Unlike physical systems information such as email can casual spawn into two (like cell mitosis) without any loss or degradation to the parent body. To address this we have the copy axion that is x < x+x
- Abstract Examples
  - Booleans: getting from a to b is a true/false question,
  - If a preorder has symmetric monoidal structure so does its opposite

Monoidal monotone maps

- The order should be preserved irrespective of when you go from P to Q (regular monotone map ref:)

Definition 2.41. Let $\mathcal{P} = (P, \leq_P, I_P, \otimes_P)$ and $\mathcal{Q} = (Q, \leq_Q, I_Q, \otimes_Q)$ be monoidal preorders. A *monoidal monotone* from $\mathcal{P}$ to $\mathcal{Q}$ is a monotone map $f\colon (P, \leq_P) \to (Q, \leq_Q)$, satisfying two conditions:

(a) $I_Q \leq_Q f(I_P)$, and

(b) $f(p_1) \otimes_Q f(p_2) \leq_Q f(p_1 \otimes_P p_2)$

for all $p_1, p_2 \in P$.

There are strengthenings of these conditions that are also important. If f satisfies the following conditions, it is called a *strong monoidal monotone*:

Bool	Cost
true	0
false	inf

- g: Bool ⟶ Cost, if it were a regular monotone mao we would not allow this true> false but 0 is not greater than inf, so the order breaks. Here we check true intx false, and 0+inf which results in a map of the output of the operations rather than the elements themselves.

Enrichment

- V categories: Let V be a symmetric monoidal preorder so. V category has 2 components. Read as X is enriched in V.
  - Ob(X) which specifies objects in X
  - X(x,y) specifies a homomorphic object
  - I ≤ $X(x, x)$
  - $X(x, y) \otimes X(y, z) \leq X(x, z)$
- Preorders as a bool category;

  due to the transistive nature of ≤, $X(x, y) \otimes X(y, z) \leq X(x, z)$ holds nicely
- Preorder as Cost category

  Let us define d(x,y) as a distance function (actual distance, effort, getting from anywhere in A just into B, nearest point) 4 requirements of the metric

space are

- d(x,x) =0
- d(x,y)= 0 implies x =y
- d(x,y)=d(y,x)
- d(x,y)+d(y,z) ≥ d(x,z)

If you drop off 2 and 4 you get the Cost category aka Lawevere Metric space

- V-variations on preorders and metric spaces
  - eg: MNY (maybe, yes and no) Let *M* be a set and let

    M =(P(*M*), ⊆, *M*, ∩) be the monoidal preorder whose elements are subsets of *M*.

Constructions on V-categories

- **Inverse**
  - So far we have seen that the V enrichment is like constructing a monotone map to the monoid. The first map is the base map and the map used for enrichment is the second one. We can now attempt to invert this process
- **Enrichment**
  - A V-category X is a *dagger* V-category if the identity function is a V-functor †: X → Xop.
    And a
    skeletal V-category is one in which if $I \leq X(x, y)$ and $I \leq X(y, x)$, then $x = y$.
- **Product V categories**
  - (i) Ob(X × Y) := Ob(X) × Ob(Y),
    (ii) (X × Y)((
    x, *y*), (*x*′, *y*′))= X(*x*, *x*′) ⊗ Y(*y*, *y*′),

    for two objects (*x*, *y*) and (*x*′, *y*′) in Ob(X × Y)

Computing presented V-categories with matrix multiplication

- **Monoidal closed preorders**

Definition 2.79. A symmetric monoidal preorder $\mathcal{V} = (V, \leq, I, \otimes)$ is called *symmetric monoidal closed* (or just *closed*) if, for every two elements $v, w \in V$, there is an element $v \multimap w$ in $\mathcal{V}$, called the *hom-element*, with the property

$$(a \otimes v) \leq w \quad \text{iff} \quad a \leq (v \multimap w). \tag{2.80}$$

for all $a, v, w \in V$.

- **Quantales**

Definition 2.90. A *unital commutative quantale* is a symmetric monoidal closed preorder $\mathcal{V} = (V, \leq, I, \otimes, \multimap)$ that has all joins: $\bigvee A$ exists for every $A \subseteq V$. In particular, we often denote the empty join by $0 := \bigvee \varnothing$.

- "One can personify the notion of unital, commutative quantale as a kind of navigator. A navigator is someone who understands "getting from one place to another." Different navigators may care about or understand different aspects—whether one can get from
  A to *B*, how much time it will take, what modes of travel will work, etc.—but they certainly have some commonalities. Most importantly, a navigator needs to be able to read a map: given routes *A* to *B* and *B* to *C*, they understand how to get
  a route
  A to *C*. And they know how to search over the space of way-points to get from *A* to *C*. These will correspond to the monoidal product and the join operations, respectively." similar to what we saw in the subjcetive decisions of stratification, you may decide what but no violate fundamental properties of how

Example 2.102. Let $\mathcal{V} = \mathbf{Bool}$. Here is an example of matrix multiplication $M * N$. Here $X = \{1,2,3\}$, $Y = \{1,2\}$, and $Z = \{1,2,3\}$, matrices $M : X \times Y \to \mathbb{B}$ and $N : Y \times Z \to \mathbb{B}$ are shown to the left below, and their product is shown to the right:

$$\begin{pmatrix} \texttt{false} & \texttt{false} \\ \texttt{false} & \texttt{true} \\ \texttt{true} & \texttt{true} \end{pmatrix} * \begin{pmatrix} \texttt{true} & \texttt{true} & \texttt{false} \\ \texttt{true} & \texttt{false} & \texttt{true} \end{pmatrix} = \begin{pmatrix} \texttt{false} & \texttt{false} & \texttt{false} \\ \texttt{true} & \texttt{false} & \texttt{true} \\ \texttt{true} & \texttt{true} & \texttt{true} \end{pmatrix}$$