Tools for Worldmaking: Universal Concepts for Mental Reality

Drew Flieder

29 November 2022; updated 8 June 2025

Contents

Chapter 1

Introductory Statement

This work presents a series of investigations into the nature of mathematical concept-building.

Part I

Structure and Composition: Universal Concepts for Thought

Chapter 2

Structures

2.1 Foundations

SUMMARY. We establish the foundational ideas underlying the structure theory to be developed in this work.

— § —

Our current task is to construct a formalism for the concept of *structure.* Indeed, there have been many such attempts to define structure, especially in areas dealing with foundations of mathematics. Our task is to define structure in the sense that mathematicians speak of 'sets with additional structure', such as e.g. groups, topological spaces, modules, and so on. Whereas (many) categories consist of such structured sets, there is to date no category the objects of which are structures *in general.* Establishing such a 'universal' category of structures is thus the objective of this chapter. We will see that the category of structures to be defined has the property of being a topos, which allows for highly general constructions of new structures from given structures.

2.1.1 Conceptual Primitives

SUMMARY. The concept of a *relation* is discussed, as it serves as the foundation for the concept of a structure.

— § —

If we think of *structure* as the relationships inherent to some set, then we can reduce the notion of structure to the primitive concept of a *relation.* Generally speaking, for a thing x to be related to a thing y means that there is some concept R which associates x with y. For instance, the 'less than or equal to' relation $\leq$ relates x to y iff x is less than or equal to y.

Mathematically speaking, given sets A and B, a relation R is a subset of their Cartesian product, namely $R \subseteq A \times B$. An element $(a, b) \in R$ is thus an ordered pair such that the relation R holds between $a \in A$ and $b \in B$.

A special case of a relation is the situation where $R \subseteq A \times A$. Then R is called an *endorelation*, since it is a relation from a set to itself, rather than between sets. For instance, the relation $\leq$ is an endorelation on the natural numbers $\mathbb{N}$.

The generality of both the concept of a set and the concept of a relation suffices to account for the concept of structure as such. Not only can we define familiar mathematical structures such as groups, topological spaces, modules, etc., but also hitherto unconceived structures. Although the structures that we will define are incarnated from an abstract and eternal mathematical layer of reality, they also partake in a concrete and historical reality that is tied to practical activities. This is like the situation in science, where there may be many instances of vector spaces $V_1, \ldots, V_n$, yet such that all are isomorphic to $\mathbb{R}$. Essentially, the uniqueness of each of these vector spaces is determined by the context in which they are instantiated, whereas $\mathbb{R}$ is the eternal layer of reality that conditions the declaration of the existence of each V_i. We will see that the definition of a structure incorporates the relation between an abstract layer of reality, and the concrete layer of reality in which the instantiations of abstract objects exist.

2.1.2 The Category of Sets and Relations

Summary. The category of sets and relations is introduced, as it serves as the basis of the structure theory to be developed.

— § —

The category **Rel** has sets for objects and relations for morphisms. **Rel** contrasts with many other categories insofar as relations between objects are more general than the functional morphisms that hold between objects of common categories such as **Set**, **Top**, **Grp**,[1] and many others. The latter categories have morphisms that are special types of functions on the underlying sets of the objects in the category. A function is a special type of relation, defined as follows:

Definition 2.1 (Function)**.** A relation $f \subseteq A \times B$ is a function iff it satisfies the following criteria:

1. Totality: f is *total* iff for every $a \in A$, there exists an $(a, b) \in f$.

2. Functionality: f is *functional* iff $(a, b), (a, c) \in f$ implies $a = c$.

[1]These categories are, respectively, the category of sets, topological spaces, and groups.

What distinguishes **Rel** from categories whose morphisms are functional is that the morphisms of **Rel** need not be total nor functional.

Since the number of relations that exist between sets is greater than the number of functions which exist between them, **Rel** contains **Set** as a wide[2] subcategory.

Composition of relations $R : A \to B$ and $S : B \to C$ in **Rel** is the standard composition of relations, i.e.,

$$S \circ R = \{(a, c) \in A \times C : \exists b \in B((a, b) \in R \wedge (b, c) \in S)\} \subseteq A \times C. \tag{2.1}$$

Furthermore, **Rel** can be turned into a 2-category,[3] where a map $\Omega : R \Rightarrow S$ is such that $R, S : A \to B$ are relations, and $R \subseteq S$.

Having laid out the basics, we may now take a first attempt at defining what constitutes a structure.

2.2 Structures: Naive Setup

Summary. A naive formalism for dealing with structures is presented. We first define *elementary structures*, and construct some examples. Once elementary structures are defined, we will then define *compound structures*.

— § —

2.2.1 Elementary Structures

Summary. The definition of an *elementary structure* is provided. We construct three different types of elementary structures. The first is a monoid, the second a topological space, and the third a module.

— § —

The idea of an elementary structure is that we choose some set X, along with a set of relations $\{R_i : A_i \to X\}_{i \in I}$ all with codomain X, and that these relations will endow X with structure. This is analogous to how signatures are defined in mathematical logic.[4] For instance, for an abelian group G, its signature $\sigma = (+, -, e)$ establishes the following:

1. The function $+ : G \times G \to G$ is the binary operation on G.
2. The function $- : G \to G$ maps elements in G to their inverses.

[2]A wide subcategory C' of C is a subcategory that contains all the objects of C.

[3]A 2-category has arrows between arrows. In a 2-category, 0-cells are objects, 1-cells are arrows, and 2-cells are arrows between arrows. In general, an n-category has cells from 0 and n, where a k-cell is an arrow between $(k-1)$-cells.

[4]See [10, p. 4] for definition of a signature.

3. The function $e : 1 \to G$ picks out the identity element in G (where 1 is understood as a singleton set).

Thus, for a set G in **Rel**, we can obviously generate such a group structure with the set $\{+, -, e\}$ of relations, which are, in this case, functions.

The use of such signatures cannot, however, achieve the ultimate goal of our task, which is (1) to construct a universal category of structures, and (2) to provide a grammar that defines the rules for constructing new structures from given structures. To achieve such a task requires a different formalism than that found in mathematical logic. We will present the rigorous theory of structures in Section 2.3, but for now, we present the definition of an *elementary structure*.

Definition 2.2 (Elementary Structure)**.** An *elementary structure* is a pair $S = (X, R)$, where X is a set in **Rel** and $R = \{R_i : A_i \to X\}_{i \in I}$ a set of relations in **Rel**, all with codomain X. We call X the *underlying set* of S and R the *generators* of S, the idea being that the relations in R *generate* the structure S on the *underlying set* X.

This definition clearly establishes the idea of having a set with additional structure. To see this idea in action, we now provide a few example constructions.

Example 2.1. Monoids.

The axioms for a monoid are the following:

Axioms 2.1 (Monoid)**.** A monoid $\mathcal{M}$ consists of a set M along with a binary operation $\bullet : M \times M \to M$ such that the following conditions hold:

1. *Identity.* There exists an element $e \in M$ such that $e \bullet a = a \bullet e = a$ for all $a \in M$.

2. *Associativity.* For all $a, b, c \in M$, $(a \bullet b) \bullet c = a \bullet (b \bullet c)$.

All of this can be encoded as an elementary structure $\mathcal{M} = (M, F)$. We first take care of the binary operation via the function $\bullet : M \times M \to M$. Now we can define the rest.

Axiom 1. The identity axiom is taken care of by first specifying the function $e : 1 \to M$ that picks out the identity element in M. However, we also have the requirement that this element $e \in M$ must be such that for all $a \in M$, we have $a \bullet e = e \bullet a = a$. We can check that this condition holds via the set-theoretic[5] pullback

$$\begin{tikzcd}
(e \times M) \times_M (M \times e) \arrow[r, "\mathrm{pr}_2"] \arrow[d, "\mathrm{pr}_1"'] \arrow[dr, dashed, "\mathrm{Id}^e_M"] & M \times e \arrow[d, "\bullet|_{M\times e}"] \\
e \times M \arrow[r, "\bullet|_{e\times M}"'] & M
\end{tikzcd} \tag{2.2}$$

(Note that e is technically shorthand for $\{e(1)\}$, since $e(1) \in M$, and we wish to take the product of M with the singleton $\{e(1)\}$.) The condition $e \bullet a = a \bullet e$ is satisfied if $(e \times M) \times_M (M \times e)$ consists only of pairs of the form $\big((e, a), (a, e)\big)$, and the morphism Id^e_M is a bijective function where $\mathrm{Id}^e_M\big((e, a), (a, e)\big) = a$.

Axiom 2. Next we need to establish the axiom of associativity. We will see that the satisfaction of this axiom, like that of the axiom of identity, can be encoded as a set-theoretic pullback. First, define the morphism

$$\bullet(\bullet, -) : (M \times M) \times M \to M \tag{2.3}$$

that sends a pair $((a, b), c)$ to the evaluation $(a \bullet b) \bullet c$. Similarly, define the morphism

$$\bullet(-, \bullet) : M \times (M \times M) \to M \tag{2.4}$$

that sends $(a, (b, c))$ to $a\bullet(b\bullet c)$. The axiom of associativity states that for all $a, b, c \in M$, we have that $(a \bullet b) \bullet c = a \bullet (b \bullet c)$. We can check that this condition holds via the set-theoretic pullback

$$\begin{tikzcd}
\big((M \times M) \times M\big) \times_M \big(M \times (M \times M)\big) \arrow[r, "\mathrm{pr}_2"] \arrow[d, "\mathrm{pr}_1"'] \arrow[dr, dashed, "{(\bullet(\bullet,-),\ \bullet(-,\bullet))}"] & M \times (M \times M) \arrow[d, "\bullet(-{,}\bullet)"] \\
(M \times M) \times M \arrow[r, "\bullet(\bullet{,}-)"'] & M
\end{tikzcd} \tag{2.5}$$

The set $\big((M \times M) \times M\big) \times_M \big(M \times (M \times M)\big)$ should contain as a subset pairs $\big(((a, b), c), (a, (b, c))\big)$, for all $a, b, c \in M$. If it does, then this means $(a \bullet b) \bullet c = a \bullet (b \bullet c)$, and therefore the axiom of associativity holds.

[5]It is important that we state explicitly that this pullback is a set-theoretic pullback, meaning that it is a pullback in **Set**, rather than a pullback in **Rel**. This is because limits and colimits work much differently in **Rel** than they do in **Set**, and moreover **Rel** does not have limits and colimits *in general*, whereas **Set** does. For instance, a product in **Rel** is likewise a coproduct via self-duality, and such (co)products are simply disjoint unions of sets.

Thus we have established a monoidal structure $\mathcal{M} = (M, R)$. Technically, all that R is required to have is the morphism $\bullet : M \times M \to M$, since all of the axioms that a monoid is required to satisfy are determined by the behavior of the binary operation. However, we add the morphisms that establish the identity element, as well those which demonstrate that the laws of identity and associativity are satisfied, in order to be explicit. We can picture the total diagram in **Rel** that establishes the structure of $\mathcal{M}$, along with a more reduced diagram that still communicates the required information, as in Figure 2.1.

Example 2.2. Topological spaces.

There are many ways to axiomatize topological spaces, one of them being via neighborhoods, which is the approach we will take.

Let X be the underlying set of a topological space. A topology on X is defined via the following axioms:

Axioms 2.2 (Topological Space)**.** Each element $x \in X$ has a non-empty collection N_x of subsets $U \subseteq X$, called *neighborhoods* of x, such that the following conditions hold:

1. If N_x is a neighborhood of x, then $x \in N_x$.
2. If M is a subset of X and includes a neighborhood of x, then M is a neighborhood of x.
3. The intersection of two neighborhoods of x is a neighborhood of x.
4. If N_x is a neighborhood of x, then it contains a neighborhood M_x such that N_x is a neighborhood of every point in M_x.

Let's translate this axiomatization into the language of elementary structures step-by-step. There are multiple ways to do this, so the following construction is one of the many viable methods to define topological spaces as elementary structures. Let X be the underlying set of the structure $Top = (X, \tau)$, where τ is to be defined via the following constructions.

Our first step is to define the relation

$$\mathcal{N} : \mathcal{P}X \to X \tag{2.6}$$

that associates neighborhoods to elements.[6] Then it is simply a matter of checking whether $\mathcal{N}$ satisfies the above axioms.

Axiom 1. To encapsulate the first axiom, we need to ensure that $\mathcal{N}$ associates U to x only if $x \in U$. We can do this as follows. First, define the relation

$$\epsilon : \mathcal{P}X \to X \tag{2.7}$$

[6]Note that $\mathcal{P}X$ is the power set of X, i.e., the set of all subsets of X.

$$\begin{tikzcd}
M \times M \arrow[dr, "\bullet"] & e \times M \arrow[d, "\bullet|_{e\times M}"] & (e \times M) \times_M (M \times e) \arrow[l, "\mathrm{pr}_1"'] \arrow[d, "\mathrm{pr}_2"] \\
(M \times M) \times M \arrow[r, "\bullet(\bullet,-)"] & M & M \times e \arrow[l, "\bullet|_{M\times e}"'] \\
((M \times M) \times M) \times_M (M \times (M \times M)) \arrow[u, "\mathrm{pr}_1"] \arrow[r, "\mathrm{pr}_2"] & M \times (M \times M) \arrow[u, "\bullet(-,\bullet)"] & 1 \arrow[ul, "e"']
\end{tikzcd}$$

(a) The total diagram of the monoidal structure defined on M.

$$\begin{tikzcd}
M \times M \arrow[dr, "\bullet"] & & (e \times M) \times_M (M \times e) \arrow[dl, "\mathrm{Id}^e_M"'] \\
& M & \\
((M \times M) \times M) \times_M (M \times (M \times M)) \arrow[ur, "(\bullet(\bullet,-),\ \bullet(-,\bullet))"] & & 1 \arrow[ul, "e"']
\end{tikzcd}$$

(b) A reduced diagram of the monoidal structure defined on M.

Figure 2.1: The diagram in (a) depicts the total process of defining $\mathcal{M}$. However, we can simplify it as in (b).

that assigns to each $U \in \mathcal{P}X$ every element $u \in U$. This ensures that $U \in \mathcal{P}X$ relates to $x \in X$ iff $x \in U$. Now, axiom 1 is satisfied iff there's a 2-morphism $\iota : \mathcal{N} \Rightarrow \epsilon$, which means that $\mathcal{N}$ is a subrelation of ϵ, i.e., $\mathcal{N} \subseteq \epsilon$. Since every pair $(U, x) \in \epsilon$ is such that $x \in U$, it is therefore the case that every pair $(V, y) \in \mathcal{N}$ also has $y \in V$. We can see this in the following diagram:

$$\begin{tikzcd} \mathcal{P}X \arrow[r, bend left, "\mathcal{N}"{name=A}] \arrow[r, bend right, "\epsilon"'{name=B}] & X \arrow[from=A, to=B, Rightarrow, "\iota"'] \end{tikzcd} \tag{2.8}$$

Note, however, that $\iota : \mathcal{N} \Rightarrow \epsilon$ cannot be a generator in τ, since by definition an an element $t \in \tau$ must be a 1-morphism with codomain X. Fortunately we can translate ι into a series of 1-morphisms that arrive at X. Since relations can be identified with sets, and sets are the objects of **Rel**, any 2-morphism $I : (R : a \to b) \Rightarrow (S : a \to b)$ can be translated into a diagram

$$\begin{tikzcd} R \arrow[r, hook, "I"] \arrow[dr, "\mathrm{pr}_2"'] & S \arrow[d, "\mathrm{pr}_2"] \\ & b \end{tikzcd} \tag{2.9}$$

where I is an inclusion function, and pr_2^S (resp. pr_2^R) is the set-theoretic projection onto the second component.

In our case, the 2-morphism $\iota : \mathcal{N} \Rightarrow \epsilon$ is translated into the diagram

$$\begin{tikzcd} \mathcal{N} \arrow[r, hook, "\iota"] \arrow[dr, "\mathrm{pr}_2"'] & \epsilon \arrow[d, "\mathrm{pr}_2"] \\ & X \end{tikzcd} \tag{2.10}$$

Note that, when considered as a set, pr_2^ϵ (resp. $\mathrm{pr}_2^\mathcal{N}$) consists of pairs $\big((U, x), x\big)$. Since each $\big((U, x), x\big)$ canonically identifies with (U, x), there is an obvious equivalence between pr_2^ϵ (resp. $\mathrm{pr}_2^\mathcal{N}$) and ϵ (resp. $\mathcal{N}$). Thus, we can check whether axiom 1 is satisfied if we have such an inclusion $\iota : \mathcal{N} \rightarrowtail \epsilon$.

Axiom 2. We need to show that given a neighborhood N of x, a superset $M \supset N$ is also a neighborhood of x.

Let $\mathcal{N}_x : \mathcal{P}X \to X$ be the subrelation of $\mathcal{N}$ such that $(U, y) \in \mathcal{N}_x$ iff $(U, y) \in \mathcal{N}$ and $y = x$. Let $p : 1 \to \mathcal{P}X$ be a function which picks out a unique element in $\mathcal{P}X$. Then $\mathcal{N}_x \circ p(1)$ is either a singleton or the empty set. It is a singleton precisely when $p(1) \in \mathcal{P}X$ is a neighborhood of x, and empty when it isn't.

Suppose that $\mathcal{N}_x \circ p(1) \neq \varnothing$, and therefore that $p(1)$ is a neighborhood of x. We have that for any other $q : 1 \to \mathcal{P}X$ such that $p(1) \subseteq q(1)$, $\mathcal{N}_x \circ q(1) \neq \varnothing$. Thus for any $\pi : 1 \to \mathcal{P}X$ such that $\mathcal{N}_x \circ \pi(1) \neq \varnothing$, there arises the set of maps $\{p_i\}_{i\in I} : 1 \to \mathcal{P}X$ containing *all* maps p_j such that $\pi(1) \subseteq p_j(1)$. This axiom guarantees that for all $p_j \in \{p_i\}_{i\in I}$, we have $\mathcal{N}_x \circ p_j(1) \neq \varnothing$, and therefore the diagram

$$\begin{tikzcd}
1 \arrow[rr, "p_j"] \arrow[dr, dashed, "\mathcal{N}_x\circ\pi"'] & & \mathcal{P}X \arrow[dl, "\mathcal{N}_x"] \\
& X &
\end{tikzcd} \tag{2.11}$$

commutes.

In the situation where x has a smallest neighborhood M_x, then all we need is the map $\min_x : 1 \to \mathcal{P}X : 1 \mapsto M_x$, and every other neighborhood of x is derived.

Axiom 3. Here we are demonstrating that for two neighborhoods N, M of x, their intersection $N \cap M$ is a neighborhood of x.

Let $p, q : 1 \to \mathcal{P}X$ be functions such that $\mathcal{N}_x \circ p(1) \neq \varnothing$ and $\mathcal{N}_x \circ q(1) \neq \varnothing$, where $\mathcal{N}_x$ is as in the previous demonstration. There exists a map $p \cap q : 1 \to \mathcal{P}X$ such that $p \cap q(1) = p(1) \cap q(1)$, and $\mathcal{N}_x \circ p \cap q(1) \neq \varnothing$. Thus the following diagram commutes:

$$\begin{tikzcd}
1 \arrow[r, shift left=2, "p"] \arrow[r, "q"] \arrow[r, shift right=2, "p\cap q"'] & \mathcal{P}X \arrow[r, "\mathcal{N}_x"] & X
\end{tikzcd} \tag{2.12}$$

Axiom 4. The last thing we need to demonstrate is that any neighborhood N of x contains a neighborhood M of x such that N is a neighborhood of every point in M.

For a subset $A \subseteq X$, let $\mathcal{N}_A : \mathcal{P}A \to X$ denote the relation which assigns neighborhoods to each $a \in A$. In other words, $\mathcal{N}_A$ is a subrelation of $\mathcal{N}$. Note that for any $p : 1 \to \mathcal{P}X$, we have that $\mathcal{N}_A \circ p(1)$ assigns $p(1)$ to $a \in A$ iff $p(1)$ is a neighborhood of a. If every element $a \in A$ has $p(1)$ as a neighborhood, then $\mathcal{N}_A \circ p(1)$ is in bijection with A: particularly, $\mathcal{N}_A \circ p(1) = A$.

Now let $\mathcal{N}_x : \mathcal{P}X \to X$ be as in the previous discussion, i.e., the assignment of neighborhoods to the point $x \in X$. We have that for any $p : 1 \to \mathcal{P}X$ such that $\mathcal{N}_x \circ p(1) \neq \varnothing$, there exists a $q : 1 \to \mathcal{P}X$ such that

- $q(1) \subseteq p(1)$,
- $\mathcal{N}_x \circ q(1) \neq \varnothing$, and, most importantly,
- $\mathcal{N}_{q(1)} \circ p(1) = q(1)$.

This establishes that $p(1)$ is a neighborhood of every point in $q(1)$. This is demonstrated in the following diagram

$$\begin{tikzcd} 1 \arrow[r, shift left, "p"] \arrow[r, shift right, "q"'] & \mathcal{P}X \arrow[r, shift left, "\mathcal{N}_x"] \arrow[r, shift right, "\mathcal{N}_{q(1)}"'] & X \end{tikzcd} \tag{2.13}$$

where $\mathcal{N}_x \circ p(1) = \mathcal{N}_x \circ q(1)$ and $q(1) = \mathcal{N}_{q(1)} \circ p(1)$.

Thus we have demonstrated that topological spaces can be encoded as elementary structures.

Example 2.3. Modules.

We now construct mathematical modules as elementary structures.

Axioms 2.3 (Module)**.** Given a ring R and an abelian group M, a left R-module $\mathbf{M}$ consists of M along with scalar multiplication $\cdot : R \times M \to M$ such that, for $r, s \in R$ and $a, b \in M$, the following axioms are satisfied:

1. $r \cdot (a + b) = r \cdot a + r \cdot b$
2. $(r + s) \cdot a = r \cdot a + s \cdot a$
3. $(rs) \cdot a = r \cdot (s \cdot a)$
4. $1 \cdot a = a$

Note that all of these axioms can be encoded as (set-theoretic) fibered products, as was done in Example 2.1 for monoids. For instance, we encode the first axiom with the following pullback diagram:

$$\begin{tikzcd} \big(R \times (M \times M)\big) \times_M \big((R \times M) \times (R \times M)\big) \arrow[r] \arrow[d] & (R \times M) \times (R \times M) \arrow[d] \\ \big(R \times (M \times M)\big) \arrow[r] & M \end{tikzcd} \tag{2.14}$$

We can construct similar pullbacks for axioms 2 and 3. The fourth axiom is simple, since all we need is the map:

$$1_R : 1 \times M \to M : (1, a) \mapsto a. \tag{2.15}$$

What is unique about the situation with modules, however, is that there are two component structures that are constitutive of the module: namely, the abelian group M and the ring R. Nonetheless, when we refer to elements in a module $\mathbf{M}$, we mean elements in its underlying group M, so therefore the set M (forgetting its group structure) is the underlying set of the module. Because of this, we need to define the group structure on M. The relations that we require for this are all of those used in Example 2.1 for monoids, plus some additional relations that establish the axiom that elements in groups have inverses. For a group G with binary operation $* : G \times G \to G$, it is required that for every $g \in G$, there exists a $g^{-1} \in G$ such that $g * g^{-1} = g^{-1} * g = e$, where e is the identity element. Thus, let $+ : M \times M \to M$ be the binary operation on M. We have the endorelation

$$- : M \to M : a \mapsto -a, \tag{2.16}$$

assigning each element in M to its inverse. Then there is the requirement that $a + (-a) = -a + a = e$, which can be encoded in a number of ways. One way is as the fibered product

$$\begin{tikzcd} M \times_M M \arrow[r, "\mathrm{pr}_2"] \arrow[d, "\mathrm{pr}_1"'] \arrow[dr, dashed, "+"] & M \arrow[d, "\mathrm{Id}_M"] \\ M \arrow[r, "-"'] & M \end{tikzcd} \tag{2.17}$$

which is such that $+ : M \times_M M \to M$ maps every pair (m, n) to the identity element $e \in M$.

We are essentially done in defining the module $\mathbf{M}$. However, for many axioms we used the set R without defining any ring structure on R. This is not really an issue, however, since when we define a module we are generally only interested in how R *acts on* M, and R's action is encoded via the map $\cdot : R \times M \to M$. Furthermore, the ring structure on R will be implicit in the construction of $\mathbf{M}$. For instance, in defining the third axiom, we would want some fibered product

$$\begin{tikzcd} \big((R \times R) \times M\big) \times_M \big(R \times (R \times M)\big) \arrow[r, "\mathrm{pr}_2"] \arrow[d, "\mathrm{pr}_1"'] & R \times (R \times M) \arrow[d, "\cdot(\cdot,-)"] \\ (R \times R) \times M \arrow[r, "\cdot(*,-)"'] & M \end{tikzcd} \tag{2.18}$$

where the morphism $\cdot(*, -) : (R \times R) \times M \to M$ implicitly encodes ring multiplication $* : R \times R \to R$. The idea of the morphism $\cdot(*, -)$ is that it will take a pair $\big((r, s), a\big)$ and apply first $r * s = t$, and then $t \cdot a$.

Encoding axiom 2 in a similar way would implicitly encode ring addition $+ : R \times R \to R$.

We'd have again the fibered product

$$\begin{tikzcd}
\big((R \times R) \times M\big) \times_M \big((R \times M) \times (R \times M)\big) \arrow[r, "\mathrm{pr}_2"] \arrow[d, "\mathrm{pr}_1"'] & (R \times M) \times (R \times M) \arrow[d, "+(\cdot,\cdot)"] \\
(R \times R) \times M \arrow[r, "\cdot(+,-)"'] & M
\end{tikzcd} \tag{2.19}$$

where $\cdot(+,-)$ again takes a pair $\big((r,s),a\big)$ and applies first $r+s=t$, and then $t \cdot a$.

Note that the specification of a ring R only depends on the specification of its additive and multiplicative operators $+$ and $*$, since all other ring axioms of R are totally determined by $+$ and $*$. Since $+$ and $*$ are implicitly encoded in the construction of $\mathbf{M}$, then we have implicitly encoded the ring structure of R.

These examples make clear how classic mathematical structures can be encoded as elementary structures. However, a common situation in mathematics is to construct new structures from given structures. Conceiving of how to do this in the context of elementary structures is thus the topic now to which we turn.

2.2.2 Compound Structures

Summary. The intuition for constructing compound structures from basic structures is discussed.

— § —

In many situations we would like to construct new structures from given structures. In type theory, such construction schemes are called *type constructors*.[7] Some common type constructors are the following:

1. *Product types.* Given objects A and B, we can form the product type $A \times B$. If A and B are sets, for example, then their product $A \times B$ is the Cartesian product consisting of all pairs (a,b) where $a \in A$ and $b \in B$.

2. *Coproduct types.* We can also form the coproduct type $A+B$. If A and B are sets, then $A+B$ is the disjoint union of A and B.

3. *Power types.* The power type $\Omega(A)$ consists of all subobjects of A. If A is a set, then $\Omega(A)$ is the usual power set construction $\mathcal{P}(A)$.

4. *Function types.* Given objects A and B, we can form the type B^A of functions $f : A \to B$. If A and B are sets, then B^A is the set of functions from A to B.

[7] See for instance [11, p. 940] for formal discussion.

In general, however, one cannot apply such type constructors to any kind of objects. Instead, the objects must behave in a certain way in order for such type constructors to be applicable. For instance, sets can accomodate all of the aforementioned constructions, but vector spaces cannot. If we are given a vector space V, we cannot in general construct the power object $\Omega(V)$.

However, we would like to carry these kinds of type constructors into the context of structures. Intuitively, we would like each of the above type constructors to correspond to the following kinds of constructions of structures:

1. *Product structures.* For two structures $S = (A, R)$ and $T = (B, Q)$, the product $S \times T$ would consist of the Cartesian product $A \times B$ on the underlying sets, and such that the A coordinate would inherit the structure given by R, whereas the B coordinate would inherit the structure given by Q. For instance, if R generates a total order on A and Q generates a group structure on B, then a pair $(a, b) \in A \times B$ consists of an order position a *and* a group element b.

2. *Coproduct structures.* Using the same structures S and T, their coproduct $S + T$ would consist of the disjoint union of A and B, where the A portion of $A + B$ has the structure generated by R and the B portion has the structure generated by Q. Again, assuming that S is a group and T a total order, then an element $k \in A + B$ is *either* an order position *or* a group element.

3. *Power structures.* For a structure S, a power structure $\Omega(S)$ would consist of all substructures of S. What it means for a structure to be a substructure of another structure will be made precise when we move to the formal setup in Section 2.3. Intuitively, one may think that $\Omega(S)$ corresponds to the power set $\mathcal{P}(A)$ of the underlying set of S, where each subset $X \subseteq A$ has the structure of S restricted to the X portion of A. However, we will see in the formal setup that the situation is even more general than this.

4. *Function structures.* For structures S and T, the function structure T^S corresponds to the set of functions $f : A \to B$ on their underlying sets, but with the intuition that we are mapping the structure of S to the structure of T. For instance, assuming that S is a total order and T a group, then a mapping $F : S \to T$ is like picking out a sequence of group elements from T.

While these descriptions of structure constructions make sense intuitively, we cannot, in general, achieve such constructions in the context of the naive formalism for structures discussed above. To be able to apply such constructors to structures will require us to move to the formal setup in Section 2.3, which will furnish us with a category of structures. We will see that this category has the unique property of being a topos, and thus accomodates all of the aforementioned type constructors.

2.3 Structures: Formal Setup

SUMMARY. The rigorous theory of structures is presented. After establishing some technical preliminaries, we provide the formal definition of a structure. Once the formal setup is provided, we will see that such structures furnish a category that consists of all structures.

— § —

We now move on to the formal theory of structures. The precursor of this theory is that developed by Mazzola in [15]. In that text, Mazzola defines what he calls a *form*, which is a generalization of a mathematical module. Although such forms are highly general objects that accommodate a wide diversity of theoretical constructions, as demonstrated by the plethora of Mazzola's theories, they are not general enough to account for structure as such. Thus, the following structure theory provides a necessary step in the direction of generality, allowing for a broader range of theoretical competence.

In format, the following formal definition of a structure is nearly identical to the formal definition of a form in [15, p. 63]. However, they are totally distinct kinds of objects, and must be treated as such. Moreover, structures generalize forms, so it is possible to transport Mazzola's theories that are based on forms into theories that are based on structures.

2.3.1 Technical Preliminaries

SUMMARY. We present the technical foundations on which the theory of structures are built. This will take us from the naive formalism of elementary structures to a rigorous formalism that can accomodate more complex constructions.

— § —

2.3.1.1 Locally Small Categories

SUMMARY. The concept of a *locally small category* is introduced.

— § —

Let $\mathscr{C}$ be a category. For objects A and B in $\mathscr{C}$, we denote by $\mathrm{Hom}_{\mathscr{C}}(A, B)$ the collection of morphisms $f : A \to B$ in $\mathscr{C}$. The collection $\mathrm{Hom}_{\mathscr{C}}(A, B)$ is called a *hom-collection.*

Now it may be the case that a hom-collection is not a set, since it is too large. This is why, in general, we speak of hom-*collections* rather than hom-*sets*. We say that $\mathscr{C}$ is *locally small* if all hom-collections in $\mathscr{C}$ are sets. This of course implies that the hom-collections exist in the category **Set**.

We now provide a proposition that will be important later.

Proposition 2.1. *The category* **Rel** *is locally small.*

Proof. For any two sets A and B in **Set**, their product $A \times B$ exists in **Set**. We also have that for any two sets X and Y, the exponential Y^X exists in **Set**. Therefore **Set** is locally small.

Since the collection of relations between sets A and B corresponds to the exponential object $2^{A\times B}$ in **Set**, we have that $\mathrm{Hom}_{\mathbf{Rel}}(A, B) \xrightarrow{\sim} 2^{A\times B}$. □

2.3.1.2 Contravariant Hom-Functors

Summary. We present the notion of a *contravariant hom-functor*, and discuss its relationship to presheaves.

— § —

Let $\mathscr{C}$ be a locally small category and X an object in $\mathscr{C}$. We define a special kind of functor from $\mathscr{C}$ into **Set**.

Definition 2.3 (Contravariant Hom-Functor)**.** A *contravariant hom-functor* is a functor

$$\mathrm{Hom}_{\mathscr{C}}(-, X) : \mathscr{C} \to \mathbf{Set}, \tag{2.20}$$

defined as follows:

1. $\mathrm{Hom}_{\mathscr{C}}(-, X)$ maps each object A in $\mathscr{C}$ to the hom-set $\mathrm{Hom}_{\mathscr{C}}(A, X)$.
2. $\mathrm{Hom}_{\mathscr{C}}(-, X)$ maps each morphism $f : A \to B$ in $\mathscr{C}$ to the function

$$\mathrm{Hom}_{\mathscr{C}}(f, X) : \mathrm{Hom}_{\mathscr{C}}(B, X) \to \mathrm{Hom}_{\mathscr{C}}(A, X) : g \mapsto g \circ f. \tag{2.21}$$

A generalization of a contravariant hom-functor is what is called a *set-valued presheaf.* Such a presheaf is a contravariant functor into **Set**. Contravariant hom-functors are thus special kinds of presheaves that are called *representable presheaves.* We call such a presheaf *representable* because the idea is that we are *representing* X via the network of *all* perspectives (morphisms) into X. We thus replace X in $\mathscr{C}$ with a functor the objects of which are sets of morphisms into X, the idea being that such objects of morphisms provide partial perspectives of X, and that X is *equivalent* to all the perspectives of X.

However, we may have only *some* morphisms into X. In that case, we do not have a representable presheaf, but we still do have a presheaf. The idea of there being partial networks of perspectives into objects is at the heart of the structure theory. Thus we have the following

Principle 2.1. *Insofar as collections of perspectives correspond to objects, different collections of perspectives give rise to different objects.*

2.3.1.3 The Yoneda Embedding

Summary. The Yoneda embedding is presented.

— § —

Let $[\mathscr{C}^{\mathrm{op}}, \mathbf{Set}]$ be the functor category of presheaves on $\mathscr{C}$. Assigning each object A in $\mathscr{C}$ to its representable presheaf $\mathrm{Hom}_{\mathscr{C}}(-, A)$ thus extends to a functor $Y : \mathscr{C} \to [\mathscr{C}^{\mathrm{op}}, \mathbf{Set}]$. The Yoneda lemma[8] implies that this functor is full and faithful, and hence Y embeds $\mathscr{C}$ inside its category of presheaves. Thus we may essentially replace $\mathscr{C}$ with its category of presheaves, without any loss of information. Moreover, the category of presheaves will often have particularly nice features that $\mathscr{C}$ does not have by itself, such as limits, colimits, a subobject classifier, and so on.

In our case, we will be using the category of set-valued presheaves over $\mathbf{Rel}$. For each object X in $\mathbf{Rel}$, we denote its representable presheaf $\mathrm{Hom}_{\mathbf{Rel}}(-, X)$ by $@X$, and we denote the hom-set $\mathrm{Hom}_{\mathbf{Rel}}(A, X)$ by $A@X$. Furthermore, we denote the presheaf category $[\mathbf{Rel}^{\mathrm{op}}, \mathbf{Set}]$ by $\mathbf{Rel}^{@}$.[9]

The idea is that elementary structures will be replaced by presheaves in $\mathbf{Rel}^{@}$. But how does this work? The reason is that in our definition of an elementary structure $S = (X, R)$, we choose a set X and a collection of relations $R = \{R_i : A_i \to X\}$. Now it is the case that such a set R induces a presheaf $P_R : \mathbf{Rel} \to \mathbf{Set}$. To see how, we first introduce the notion of a sieve.

Definition 2.4 (Sieve)**.** Let $\mathscr{C}$ be a category and X an object in $\mathscr{C}$. A *sieve* C on X is a collection of morphisms with codomain X that are closed under precomposition with morphisms in $\mathscr{C}$. In other words, C is a collection of morphisms such that whenever $(f : A \to X) \in C$ and $(g : B \to A) \in \mathscr{C}_1$, then $(f \circ g : B \to X) \in C$.

In [14, p. 38] it is demonstrated that the collection of sieves on X is in one-to-one correspondence with the collection of subfunctors of $\mathrm{Hom}_{\mathscr{C}}(-, X)$. Thus for each sieve C on X we have a subfunctor $\widehat{C} \subseteq \mathrm{Hom}_{\mathscr{C}}(-, X)$. Hence for an elementary structure $S = (X, R)$, R induces a sieve on X via the closure of R under precomposition of morphisms in $\mathbf{Rel}$. Therefore we get the corresponding subfunctor of $@X$. Since we will often be starting with an elementary structure $S = (X, R)$ and deriving its respective presheaf in $\mathbf{Rel}^{@}$, we define the map

$$\mathrm{PSh} : \bigcup_{X \in \mathbf{Rel}_0} \mathcal{P}\left(\bigcup_{A \in \mathbf{Rel}_0} \mathrm{Hom}_{\mathbf{Rel}}(A, X) \right) \to \mathbf{Rel}^{@}. \tag{2.22}$$

The domain consists of the following. For each set X, we union over all of the hom-sets that have X as codomain, and then take the power set. This way, we have all subsets of

[8]See [13, 59–62] for detailed discussion on the Yoneda lemma.
[9]This is inheriting the notation used by Mazzola in [17].

relations into X, which is what we need to define an elementary structure on X. Then we simply do this for every set X in **Rel**. We map the resulting sets of relations in the domain to their respective presheaves in $\mathbf{Rel}^{@}$.

2.3.2 Formal Definition of a Structure

Summary. The formal definition of a structure is provided, along with discussion of the parameters of a structure.

— § —

Definition 2.5 (Structure)**.** A *structure* S is a quadruple $S = (N, T, C, I)$, where:

1. N is the *name* of K; it consists of a string of symbols from the free monoid $\mathfrak{N}$ over some alphabet A. The alphabet can be thought to consist of all symbols from all languages, both formal and informal, to allow for maximal freedom when it comes to naming. We refer to the name of S via Name(S), or if no confusion is likely then simply N(S).
2. T is the *type* of S, and is one of the following symbols:
   (a) **Simple**,
   (b) **Limit**,
   (c) **Colimit**,
   (d) **Power**,
   (e) **Sub**,
   (f) **Hom**.

   We refer to the type of S via Type(S), or if no confusion is likely then simply T(S).
3. C is the *coordinator* of S, and it depends on the type T as follows:
   (a) If T is **Simple**, then C is a set X.
   (b) If T is **Limit** or **Colimit**, then C is a diagram[10] $\mathcal{D}$ of structures.
   (c) If T is **Power** or **Sub**, then C is a structure.
   (d) If T is **Hom**, then C is an ordered pair (C_1, C_2) of structures.

   We refer to the coordinator of S via Coordinator(S), or if no confusion is likely then simply C(S). If T is **Limit**, **Colimit**, or **Hom**, then the component structures of the coordinator are called the *coordinator structures*, or just *coordinators* for short.

[10] A diagram $\mathcal{D} : J \to \mathscr{C}$ is a functor, where J is a (very) small index category. The idea is that a diagram picks out a collection of objects and morphisms in $\mathscr{C}$.

4. I is the *identifier* of S, and is a monomorphism of functors $I : Fu \rightarrowtail A$ in $\mathbf{Rel}^{@}$. The codomain A is defined as follows:

   (a) If T is **Simple**, then $A = @X$.
   (b) If T is **Limit**, then $A = lim(\mathcal{D})$.
   (c) If T is **Colimit**, then $A = colim(\mathcal{D})$.
   (d) If T is **Power**, then $A = \Omega^{Fun(C)}$.
   (e) If T is **Sub**, then $A = Fun(C)$.
   (f) If T is **Hom**, then $A = C_2^{C_1}$.

   The point of the identifier is that it associates the functor Fu of S with the functor given by the type and coordinator. This is important especially for when T is **Sub**, as it enables us to define a structure as a substructure Fu of some larger structure A in which it lives. In general, we denote the codomain of the identifier by $Fun(C)$, which is the functor of the coordinator. The domain Fu of the identifier is what we call the functor of S. We refer to the identifier of S via Identifier(S), or if no confusion is likely then simply I(S).

We denote a structure via the notation

$$\text{Name} \underset{I:Fu \rightarrowtail A}{\longrightarrow} \text{Type(Coordinator)}. \tag{2.23}$$

We now provide the intuition for each parameter of a structure.

1. *Name.* Structures are provided with names for semiotic reasons. The name acts as a *signifier* for the structure which is the *signified.* As we will see, the *signification* – which is the process which relates the signifier to the signified – is provided by the identifier parameter.

2. *Type.* The type of a structure expresses the construction rule to be applied to the coordinator. The types are described as so:

   (a) Structures of type **Simple** are the most basic. They correspond precisely to elementary structures.
   (b) The **Limit** type generalizes the product type that we discussed above in Section 2.2.2. Specifically, consider the following. Suppose we have a diagram $\mathcal{D}$ of structures, as so

$$\begin{tikzcd} S_1 \arrow[r, "f"] & S_2 & S_3 \arrow[ll, bend left, "g"] \end{tikzcd}$$

The resulting limit $lim(\mathcal{D})$ will be a subobject of the product $S_1 \times S_2 \times S_3$, consisting of those triples $(a, b, c) \in S_1 \times S_2 \times S_3$ that satisfy the following constraints:

$$\begin{array}{rcll} a & = & g(c) & \\ b & = & f(a) & = f(g(c)) \\ c & = & c & \end{array} \tag{2.24}$$

Therefore limits generalize products by allowing constraints to hold between the coordinates. A product is thus a limit whose coordinator is a discrete diagram of structures.

(c) Likewise, the **Colimit** type generalizes the coproduct type. Although colimits are categorically dual to limits, their concrete realization is much different. Consider now the colimit of the previous diagram

$$\begin{tikzcd} S_1 \arrow[r, "f"] & S_2 & S_3 \arrow[ll, bend left, "g"] \end{tikzcd}$$

What the colimit $colim(\mathcal{D})$ determines is an equivalence relation $\sim$ on the disjoint union $S_1 + S_2 + S_3$, where the equivalence relation is determined by f and g. Specifically, for elements $a \in S_1, b \in S_2, c \in S_3$, we have:

$$\begin{array}{rcl} a \sim b & \text{iff} & f(a) = b \\ c \sim a & \text{iff} & g(c) = a \\ c \sim b & \text{iff} & f(g(c)) = b \end{array} \tag{2.25}$$

Thus $colim(\mathcal{D}) = (S_1 + S_2 + S_3)/\sim$. As with limits, a coproduct is a colimit whose coordinator is a discrete diagram of structures.

(d) The **Power** type constructs the set[11] of all substructures of its coordinator structure. Specifically, a **Power** structure with coordinator A consists of all the subfunctors of $Fun(A)$.

(e) The **Sub** type constructs a subobject of its coordinator structure. The **Sub** type is, from a formal standpoint, unnecessary. For instance, let

$$\sigma \underset{Fu \mapsto @X}{\longrightarrow} \mathbf{Simple}(X)$$

[11]The fact that **Rel** is large may lead one to be skeptical of this type constructor, since it is generally considered that the source category $\mathscr{C}$ of the presheaf category $[\mathscr{C}^{\mathrm{op}}, \mathbf{Set}]$ should be small, as discussed e.g. in [14]. There are a number of rejoinders to this bureaucratic problem. One is that we may think in terms of non-well-founded sets, so that we do not distinguish between small and large sets. Another point is that any standard structure in mathematics will be determined by an elementary structure whose set of generators is a small set, so it seems reasonable to think of structures as being determined by a small amount of data. In the worst case scenario, one can simply take a small subcategory of **Rel**, and derive therefrom a plethora of mathematical structures.

be a structure. We can get a subobject of σ via

$$\Sigma \underset{Gu \rightarrowtail Fu}{\longrightarrow} \mathbf{Sub}(X),$$

where Gu is a subfunctor of Fu. However, we could have just defined this structure in the first step, via

$$\Sigma' \underset{Gu \rightarrowtail @X}{\longrightarrow} \mathbf{Simple}(X).$$

Nonetheless, the **Sub** type is important for practical reasons involving history. We may wish to construct a structure S as a stepping stone to a more restricted subobject S' of S. Such intermediate constructions are ubiquitous in practice, and we would not be able to achieve them without the **Sub** type.

(f) The **Hom** type constructs the collection of structure morphisms $f : C_1 \to C_2$, where C_1 and C_2 are the first and second coordinates, respectively, of the coordinator C.

3. *Coordinator.* The coordinator is the 'meat' of a structure. It consists of the lower level object(s), whereas the type is the rule for constructing a new object from these lower level object(s).

4. *Identifier.* As is explained in the definition, the identifier associates the functor of the structure with the functor given by the type and coordinator of the structure. For instance, if we have the structure

$$S \underset{Fu \rightarrowtail Fun(A \times B)}{\longrightarrow} \mathbf{Limit}(A, B),$$

where the coordinator (A, B) is a discrete diagram, then the identifier associates Fu to a subobject of the functor $Fun(A \times B)$. One of the main virtues of the identifier is how it incorporates the distinction between concrete and abstract structures. This is because the codomain A of the identifier is an object in $\mathbf{Rel}^{@}$, and the instantiation of a structure S is a concrete object that is identified with (a subobject of) A. This is like how in a computer program, we may assign the same value to two variables, e.g. as in '$x = 2$' and '$y = 2$'. If we view such objects at a proper level of abstraction, we see that although it is the case that the value of x is equal to the value of y, it is not the case that the entire object '$x = 2$' is equal to '$y = 2$'. They are isomorphic, but not equal. Analogously, if we have two structures, such as e.g.

$$S \underset{Fu \rightarrowtail @X}{\longrightarrow} \mathbf{Simple}(X)$$

and

$$T \underset{Fu \rightarrowtail @X}{\longrightarrow} \mathbf{Simple}(X),$$

then we can say that S and T are equivalent, since they have the same identifier monomorphism, but not equal, since the difference of their names corresponds to there being two concrete instantiations of the abstract structure given by the identifier $Fu \rightarrowtail @X$.

We thus get the category **Str** of structures where:

1. The objects are structures.

2. A morphism $\varphi : S \to T$ is a natural transformation of their underlying functors in $\mathbf{Rel}^{@}$.

A last note is in order before moving on. In the context of forms in [15], there can both regular and circular structures. Mazzola thus presents a definition,[12] which we present here but with 'form' replaced by 'structure':

Definition 2.6 (Regular and Circular Structures)**.** Let ω be an ordinal number.

1. A structure is *regular of level* $\omega = 0$ iff it is of type simple.

2. Suppose that regular structures of all levels $\mu < \omega$ have been defined. Then a structure is *regular of level* ω iff all its coordinator structures are regular of levels $\mu < \omega$, and if ω is the successor of all the coordinators' levels.

3. A structure is *regular* iff there is an ω such that the form is regular of level ω.

4. A structure is called *circular* iff it is not regular.

An obvious example of a circular structure S is one which contains S as a coordinator, for instance

$$S \underset{f:Fu \rightarrowtail Fu \times Fun(T)}{\longrightarrow} \mathbf{Limit}(S, T).$$

Such structures are called circular for the obvious reason that they contain themselves.

2.3.3 $\mathbf{Rel}^{@}$ as a Universal Concrete Category

Summary. We contend that $\mathbf{Rel}^{@}$ furnishes a universal concrete category – that is, a category consisting of *all* the structured sets that can be thought to exist.

— § —

[12]Ibid., 76.

A *concrete category* $\mathscr{C}$ is a category such that there is a faithful forgetful functor $U : \mathscr{C} \to \mathbf{Set}$.[13] The idea is that $\mathscr{C}$ consists of objects that are sets with additional structure. Thus the forgetful functor U assigns each object X in $\mathscr{C}$ to the underlying set of X.

Since I contend that $\mathbf{Rel}^{@}$ consists of all possible structured sets, I therefore make the following

Conjecture 2.1. *Every concrete category embeds fully faithfully into* $\mathbf{Rel}^{@}$.

This means that $\mathbf{Rel}^{@}$ is a universal concrete category in the sense that every concrete category corresponds to a subcategory of $\mathbf{Rel}^{@}$. The next two sections (2.3.3.1–2.3.3.2) justify this conjecture.

2.3.3.1 The Collection of Structures on a Given Set

Summary. We discuss the technical machinery that provides all of the structures on a given set.

— § —

For $\mathscr{C}$ a locally small category, the subobject classifier of $[\mathscr{C}^{\mathrm{op}}, \mathbf{Set}]$ is a Heyting algebra. Thus, for a representable presheaf $@X$ in $\mathbf{Rel}^{@}$, the poset of subobjects $Sub(@X)$ is a Heyting algebra, consisting of all subfunctors of $@X$. What $Sub(@X)$ consists of is therefore the collection of *all* structures that can be defined on the set X. If any kind of structured set can be encoded as an object in $\mathbf{Rel}^{@}$, as we indeed claim, then for any concrete category $\mathscr{C}$, an object $\mathcal{X}$ in $\mathscr{C}$ whose underlying set is X will correspond to a functor in $Sub(@X)$.

We denote by $\preceq$ the partial order relation on $Sub(@X)$. For $S, T \in Sub(@X)$ we have that $S \preceq T$ iff S is a subfunctor of T. If S is a subfunctor of T, then this means that the set of relations on X given by S is a subset of the set of relations on X given by T. Generally speaking, if $S \preceq T$, then this means that the structure generated by T is more *specific* than the structure generated by S. Contrarily, the structure generated by S is more *general* than the structure generated by T. This is best demonstrated by an example. Suppose that S generates a monoid structure on X, as given by the kinds of relations from Example 2.1. We know that monoids generalize groups, insofar as a monoid is a just a group that doesn't necessarily have inverses. Thus we should obtain a group structure T on X simply by adding more relations to S. Specifically, we must add those relations which establish the group axioms involving inverses. This means that the relations from T would constitute a superset of the relations from S, and therefore $S \preceq T$ in $Sub(@X)$. This clearly demonstrates that the order relation $\preceq$ corresponds to a relation of specialization/generalization on structures. We can therefore say that if $S \preceq T$, then S is a *generalization* of T, whereas T is a *specialization* of S.

[13][13, p. 26].

The poset $Sub(@X)$ thus comes equipped with two closure operators. For a functor $S \in Sub(@X)$, we have the closure operator

$$\mathrm{cl}\downarrow(S) = \big\{\underline{S} : \underline{S} \preceq S\big\}, \tag{2.26}$$

which gives the set of all generalizations of S. Likewise, we have the closure operator

$$\mathrm{cl}\uparrow(S) = \big\{\bar{S} : \bar{S} \succeq S\big\}, \tag{2.27}$$

which gives the set of all specializations of S.

2.3.3.2 Structure Morphisms

Summary. The intuition of a structure morphism is presented. Structure morphisms are an essential component of the composition theory that is to be developed in the sequel.

— § —

Our objective in this section is to gain some intuition for the behavior of structure morphisms. Specifically, we will now demonstrate the behavior of structure morphisms between structures of type simple, so for the remainder of this section assume that the structures that we speak of are of type simple. Also, since structure morphisms correspond to morphisms on the functors of structures, we can forget about structure names in what follows, and assume that structures are objects in $\mathbf{Rel}^{@}$.

In the following, assume that X and Y are the underlying sets of S and T, respectively. Although a structure S is a functor, it corresponds to a structured *set*, so by abuse of notation we can write $x \in S$ to express that x is an element in the underlying set of S (soon we will see that we replace such elements x with certain morphisms that are indeed in S). Thus the idea of a structure morphism $f : S \to T$ is that f associates *structured* elements $x \in S$ to *structured* elements $y \in T$. Since S and T are functors, the morphism f corresponds to a natural transformation. By the definition of a natural transformation, we have

1. for each A in $\mathbf{Rel}$, a morphism $f_A : S(A) \to T(A)$ in $\mathbf{Set}$, and
2. for each $R : A \to B$ in $\mathbf{Rel}$ we have $f_A \circ S(R) = T(R) \circ f_B$, expressed by the commutativity of the following diagram on the right

$$\begin{tikzcd} A \arrow[d, "R"'] & S(A) \arrow[r, "f_A"] & T(A) \\ B & S(B) \arrow[u, "S(R)"] \arrow[r, "f_B"'] & T(B) \arrow[u, "T(R)"'] \end{tikzcd} \tag{2.28}$$

To see how $f : S \to T$ maps structured points of S to structured points of T, consider the following. Suppose that S contains all functions $p : 1 \to X$, where p is a function from the singleton set 1 to an element in the underlying set X of S. The function p thus picks out an element of X, and since p is in the functor S, we can say that p picks out an element in S. We call an arbitrary morphism $R : A \to X$ in S an *A-addressed*[14] *point of S*, and if R is a function, we can call it a *functional A-addressed point of S*. Therefore functional 1-addressed points $p : 1 \to X$ of S pick out elements in the underlying set X of S, and we simply write $p \in S$. So when we say e.g. that $x \in S$, where x is in the underlying set X, what we really mean is that the function $p_x : 1 \to X : 1 \mapsto x$ is in S.

Since S is a functor, we then get

$$\begin{tikzcd}
1 \arrow[r, "S"] \arrow[d, "p"'] \arrow[dr, phantom, "\overset{S}{\Longrightarrow}"] & S(1) = \{\dots, p, \dots\} \\
X \arrow[r, "S"'] & S(X) = \{\dots, \mathrm{Id}_X, \dots\} \arrow[u, "S(p)(\mathrm{Id}_X)=\mathrm{Id}_X \circ p"']
\end{tikzcd} \tag{2.29}$$

Thus, given $p : 1 \to X$ in **Rel**, we get a corresponding morphism $S(p) : S(X) \to S(1)$ that sends the identity morphism Id_X to $\mathrm{Id}_X \circ p = p$. In other words, $S(p)$ essentially establishes that $p(1) \in X$ is also a point of S. So this enables us to think of S as a *set* with structure, and therefore to say that $p \in S$.

Next, we need to evaluate $T(p : 1 \to X)$, in order to understand the intuition behind the morphism $f : S \to T$. By the Yoneda embedding, f corresponds to a map $\mathfrak{f} : X \to Y$ in **Rel**. This determines $T(p : 1 \to X)$ as so

$$\begin{tikzcd}
1 \arrow[r, "T"] \arrow[d, "p"'] \arrow[dr, phantom, "\overset{T}{\Longrightarrow}"] & T(1) = \{\dots, \mathfrak{f} \circ p, \dots\} \\
X \arrow[r, "T"'] & T(X) = \{\dots, \mathfrak{f}, \dots\} \arrow[u, "T(p)(\mathfrak{f})=\mathfrak{f}\circ p"']
\end{tikzcd} \tag{2.30}$$

In other words, $T(p)$ sends the point p to $p \circ \mathfrak{f}$, which is equivalent to operating $\mathfrak{f}(p(1))$ in **Rel**! So f does indeed associate the *structured* points of S with the *structured* points of T, via the commutativity of the square in Figure 2.2.

It is important to note, however, that for such a structure morphism $f : S \to T$ to exist, it must be the case that the corresponding morphism $\mathfrak{f} : X \to Y$ be in T. If we would like to perform such a mapping $f : S \to T$, then we can assume that if $\mathfrak{f} \notin T$ then there is an ‘updating’ of the functor T that occurs ‘under the hood’.

It is also a good time to declare the following

[14]This is following the terminology of [15].

$$\begin{tikzcd}
X \arrow[d, "\mathfrak{f}"'] & S(1) = \{\dots, p, \dots\} \arrow[r, "f(p)=\mathfrak{f}\circ p"] & T(1) = \{\dots, \mathfrak{f}\circ p, \dots\} \\
Y & S(X) = \{\dots, \mathrm{Id}_X, \dots\} \arrow[u, "S(p)"] \arrow[r, "f(\mathrm{Id}_X)=p\circ\mathrm{Id}_X=p"'] & T(X) = \{\dots, p, \dots\} \arrow[u, "T(p)"']
\end{tikzcd}$$

Figure 2.2: The square in the figure shows how structured points of S map to structured points of T. This mapping is determined via the map $\mathfrak{f} : X \to Y$ in **Rel**.

Convention 2.1. For any structure S, unless explicitly stated otherwise, assume that S contains all functional 1-addressed points $p : 1 \to X$ of S's underlying set X.

This convention is useful for a number of reasons, one of which is that it allows us to refer to an element $p \in S$ with the intuition that we are referring to the element $p(1)$ in the underlying set X, which is such a common situation in mathematics. For instance, when we refer to an element in a module by $m \in M$, we are using set-theoretic notation even though M is a module.

Since we have established that structure morphisms correspond to morphisms between the underlying sets of the structures, it is obvious that the morphisms in concrete categories correspond to structure morphisms. This is because a concrete category $\mathscr{C}$ has a forgetful functor U that is *faithful*, and thus $\mathrm{Hom}_{\mathscr{C}}(M, N) \subseteq \mathrm{Hom}_{\mathbf{Set}}(UM, UN)$. So for any morphism $f : M \to N$ in $\mathscr{C}$ we will certainly have a corresponding set map $Uf : UM \to UN$ on their underlying sets, and this set map corresponds to a structure morphism in $\mathbf{Rel}^{@}$.

2.4 Construction Schemes of Structures

Summary. The notion of a construction scheme of a structure is presented. The idea is that a structure is, independently of its component structures, constructed via a series of construction rules. Defining such construction schemes will enable us to define an equivalence relation on structures based on their construction schemes.

— § —

2.4.1 The Graphical Representation of a Structure

Summary. We show how to represent structures graphically.

— § —

For a structure S, we can represent it graphically by a tree diagram that depicts its hierarchical construction.[15] For instance, suppose that S is defined as

$$S \underset{\mathrm{Id}}{\longrightarrow} \mathbf{Limit}(S_1, S_2, S_3).$$

Then we can represent S as a tree graph

$$\begin{tikzcd} & S \arrow[d, no head] & \\ & \textbf{Limit} \arrow[dl, no head] \arrow[d, no head] \arrow[dr, no head] & \\ S_1 & S_2 & S_3 \end{tikzcd}$$

[15]As in [15, p. 52].

However, this diagram is not complete, since the coordinator structures likewise branch out to different structures. For instance, we may have the following:

$$\begin{aligned} &S_1 \underset{\text{Id}}{\longrightarrow} \mathbf{Limit}(S_{11}, S_{12}), \\ &S_2 \underset{\text{Id}}{\longrightarrow} \mathbf{Power}(S_{21}), \\ &S_3 \underset{Fu \mapsto @X}{\longrightarrow} \mathbf{Simple}(@X). \end{aligned}$$

Likewise, the coordinators of S_1 and S_2 will likewise branch out, whereas the coordinator $@X$ of S_3 cannot branch out further. We'd thus get a graphical representation as in Figure 2.3a.

Every regular structure can therefore be graphically represented as a tree graph that terminates on sets.

Furthermore, notice that immediately below each structure is its type. In order to derive a tree diagram that contains only structures, we may remove the types in the tree. Also, since representable presheaves are not yet structures, we can remove them as well, thus deriving a tree diagram consisting only of structures. The graphical representation of S in Figure 2.3a thus simplifies to the graphical representation in Figure 2.3b. We call such a reduced graphical representation of S the *ramification tree* of S, and denote it by $\mathbb{T}_S$. We can think of a ramification tree of a structure S as a directed graph (also known as a quiver) whose initial vertex is S, and whose terminal vertices are simple structures.

2.4.2 Classification of Structures According to Construction Schemes

Summary. The category of quivers is presented as a stepping stone to the category of ramification trees. After defining the category of ramification trees, we are then able to provide an equivalence relation on structures that classifies structures that are constructed according to the same scheme.

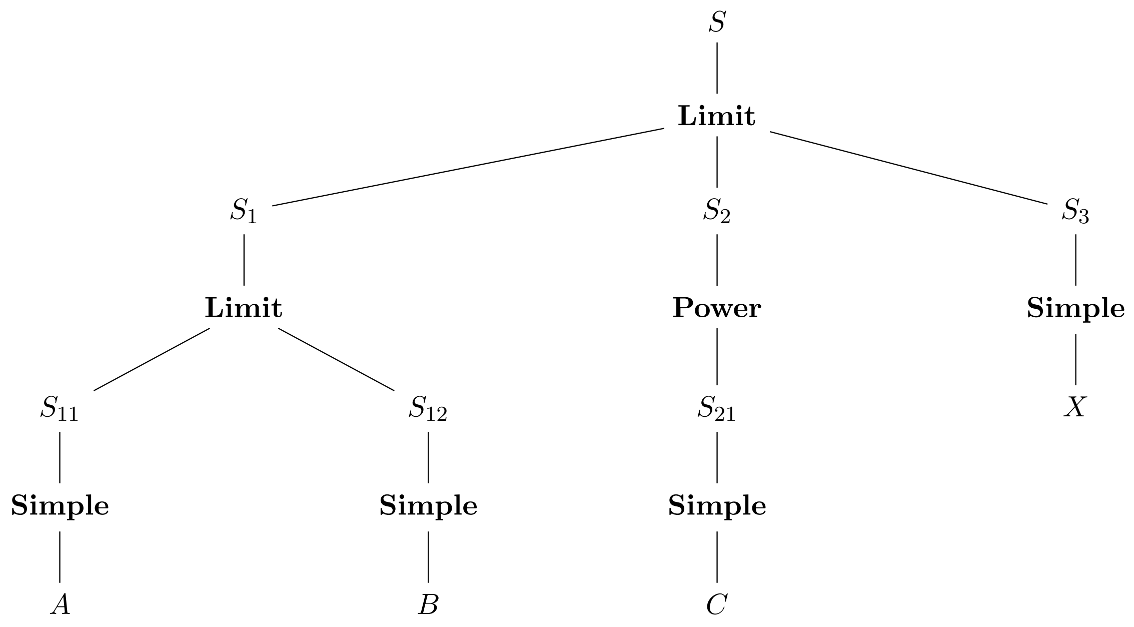
— § —

At the end of this section we will present a category of ramification trees, which will enable us to classify structures according to what will be called *constructional equivalence.* But since the category of ramification trees is a subcategory of the category of quivers, we first define quivers.

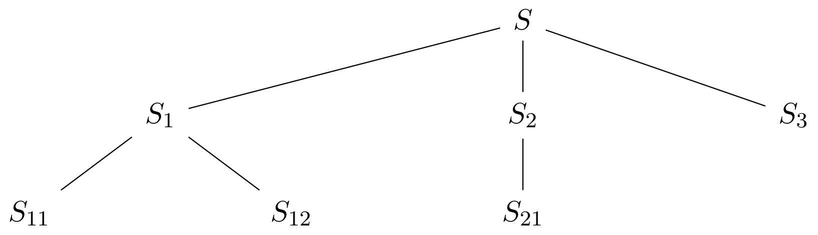
Definition 2.7 (Quiver)**.** Let X be the following category:

$$\begin{tikzcd} 0 \arrow[r, shift left, "s"] \arrow[r, shift right, "t"'] & 1 \end{tikzcd} \tag{2.31}$$

A *quiver* is a set-valued presheaf $Q : X^{\text{op}} \to \mathbf{Set}$ such that:



(a) The full graphical representation of the structure S.



(b) The reduced graphical representation of the structure S.

Figure 2.3: Two kinds of graphical representations of the structure S: (a) provides the full graphical representation, whereas (b) provides a reduced version.

1. $Q(0) = V$ is a set of vertices and $Q(1) = E$ is a set of edges.

2. $Q(s) : E \to V$ maps edges to their source vertex and $Q(t) : E \to V$ maps edges to their target vertex.

We thus get the category **Quiv** of quivers as the category of presheaves on X. We can denote a quiver as a tuple $Q = (V, E, s, t)$.

For quivers $Q = (V, E, s, t)$ and $P = (V', E', s', t')$, a morphism $\Gamma : Q \to P$ corresponds to a pair of morphisms $\gamma_0 : V \to V'$ and $\gamma_1 : E \to E'$ such that $s' \circ \gamma_1 = \gamma_0 \circ s$ and $t' \circ \gamma_1 = \gamma_0 \circ t$, as given by the following diagram

$$\begin{tikzcd}
E \arrow[r, shift left, "s"] \arrow[r, shift right, "t"'] \arrow[d, "\gamma_1"'] & V \arrow[d, "\gamma_0"] \\
E' \arrow[r, shift left, "s'"] \arrow[r, shift right, "t'"'] & V'
\end{tikzcd} \tag{2.32}$$

Thus for a ramification tree $\mathbb{T}_S$, we can recover the types of each constitutive structure of S via the isomorphism $\mathrm{Type}(\mathbb{T}_S) = \mathrm{T}(\mathbb{T}_S)$ that sends each vertex to its type. Technically, the vertex set of the codomain of T will have to be a multiset of types, in order to preserve the edge structure of the quiver, since the same type can occur at multiple different vertices. For instance, the ramification tree $\mathbb{T}_S$ in Figure 2.3b would give the following quiver $\mathrm{T}(\mathbb{T}_S)$:

$$\begin{tikzcd}
 & & & \textbf{Limit} \arrow[dll] \arrow[d] \arrow[dr] & \\
 & \textbf{Limit} \arrow[dl] \arrow[dr] & & \textbf{Power} \arrow[d] & \textbf{Simple} \\
\textbf{Simple} & & \textbf{Simple} & \textbf{Simple} &
\end{tikzcd}$$

We would next like to create a category of ramification trees. However, we first discuss a couple equivalence relations on structures. The first equivalence relation is the obvious one given by equivalence of types. That is, we have $S \sim_t S'$ if $\mathrm{T}(S) = \mathrm{T}(S')$. A much more rigorous equivalence relation would be one that takes into account the entire ramification trees of structures. We will eventually define this equivalence relation, but first we define an intermediate equivalence relation $\sim_{t'}$ as follows. Let S and S' be structures such that $\mathrm{T}(S) = \mathrm{T}(S') = \tau$, and recall that we denote the coordinator of a structure A by $\mathrm{C}(A)$. Then we define $S \sim_{t'} S'$ as so:

1. If τ is **Simple**, then $S \sim_{t'} S'$.

2. If τ is **Sub** or **Power**, then $S \sim_{t'} S'$ if $\mathrm{T}(\mathrm{C}(S)) = \mathrm{T}(\mathrm{C}(S'))$.

3. If τ is **Limit** or **Colimit**, then $S \sim_{t'} S'$ if the following conditions hold:
   (a) The coordinator structures of S and S' are in bijection.
   (b) Suppose the cardinality of the set of coordinators of S and S' is n, and let $I = \{1, \ldots, n\}$ be the set that indexes the coordinators of S and S'. Then for the symmetric group $\mathfrak{S}_n$ there exists a permutation $\pi \in \mathfrak{S}_n$ such that for every coordinator structure C_i of S and D_i of S', we have $\mathrm{T}(C_{\pi(i)}) = \mathrm{T}(D_i)$.
4. If τ is **Hom**, then $S \sim_{t'} S'$ if $\mathrm{T}(\mathrm{C}(S)_i) = \mathrm{T}(\mathrm{C}(S')_i)$ for $i \in \{1, 2\}$.

The equivalence relation $\sim_{t'}$ refines the equivalence relation $\sim_t$, since $\sim_{t'}$ also takes into account the types of the coordinators. It is obvious how $\sim_{t'}$ classifies **Simple**, **Sub**, and **Power** structures; however, understanding how $\sim_{t'}$ classifies **Limit**, **Colimit**, and **Hom** structures requires some explanation.

If τ is **Limit** or **Colimit**, then the ordering of the coordinators does not matter. For example, if we have structures

$$S \xrightarrow[\mathrm{Id}]{} \mathbf{Limit}(C, D)$$

and

$$S' \xrightarrow[\mathrm{Id}]{} \mathbf{Limit}(D, C),$$

then S and S' are isomorphic. Same for if the type is **Colimit** instead of **Limit**. Hence two (co)limit structures are equivalent by $\sim_{t'}$ if their coordinators correspond with respect to type. For instance, suppose we have structures

$$T \xrightarrow[\mathrm{Id}]{} \mathbf{Limit}(C_1, C_2, C_3)$$

and

$$T' \xrightarrow[\mathrm{Id}]{} \mathbf{Limit}(D_1, D_2, D_3),$$

such that $\mathrm{T}(C_1) = \mathbf{Colimit}, \mathrm{T}(C_2) = \mathbf{Power}, \mathrm{T}(C_3) = \mathbf{Power}$ and $\mathrm{T}(D_1) = \mathbf{Colimit}, \mathrm{T}(D_2) = \mathbf{Power}, \mathrm{T}(D_3) = \mathbf{Power}$. Then $T \sim_{t'} T'$ since T and T' both have one **Colimit** coordinator structure and two **Power** cordinator structures.

On the other hand, for **Hom** types, the ordering of the coordinators does matter. For instance, the structures

$$S \xrightarrow[\mathrm{Id}]{} \mathbf{Hom}(C, D)$$

and

$$S' \xrightarrow[\mathrm{Id}]{} \mathbf{Hom}(D, C)$$

are not in general isomorphic, since the set of morphisms from C to D is not in general isomorphic to the set of morphisms from D to C.

We defined this equivalence relation $\sim_{t'}$ because it provides some preliminary intuition for understanding the justification for the types of morphisms that exist in the category of ramification trees. We thus present this category.

Definition 2.8 (Category of Ramification Trees)**.** The category $\mathcal{T}$ of ramification trees is the subcategory of **Quiv** defined as follows:

1. Its objects are ramification trees.
2. A morphism $\Gamma : \mathbb{T}_S \to \mathbb{T}_{S'}$ of ramification trees is a monomorphism of quivers such that the following conditions hold:
   (a) For every vertex $v \in \mathbb{T}_S$, we have $\mathrm{T}(v) = \mathrm{T}(\Gamma(v))$.
   (b) If $\mathrm{T}(v) = \mathbf{Hom}$, then for v's coordinators v_1, v_2 we require $\Gamma(v_1) = w_1$ and $\Gamma(v_2) = w_2$.

Hence, a morphism $\Gamma : \mathbb{T}_S \to \mathbb{T}_{S'}$ is such that for each vertex $v \in \mathbb{T}_S$, we have $v \sim_{t'} \Gamma(v)$.

Having defined the category of ramification trees, we can now define the equivalence relation that we wanted all along, which we call *constructional equivalence.*

Definition 2.9 (Constructional Equivalence)**.** Two structures S and S' are constructionally equivalent if there is an isomorphism $\Delta : \mathbb{T}_S \to \mathbb{T}_{S'}$ in $\mathcal{T}$ on their respective ramification trees.

Thus, the isomorphism classes of ramification trees in $\mathcal{T}$ classify those structures that are constructionally equivalent to one another. Thus the skeletal category $\mathrm{sk}(\mathcal{T})$ of $\mathcal{T}$ is the category whose objects are the isomorphism classes of ramification trees.

Chapter 3

Compositions

3.1 Ontology

Summary. We present the intuitive ideas underlying the ontology of the Composition Theory to be developed in this chapter. A connection between our Composition Theory and Aristotle's hylomorphic theory of beings is made. We then outline the formal objective of this chapter, where we suggest the path that we will take to develop a universal method of encoding.

— § —

The objective of this chapter is to provide a universal method for encoding any kind of object whatsoever. We are thus tasked with constructing an ontology that is not restricted to this or that domain of knowledge, but rather the entire domain of human knowledge. I claim that the Composition Theory provides an ontology that is both *open* – meaning that it allows for the construction of hitherto unidentified objects – and *universal* – meaning that it provides the tools for rationally constructing any object whatsoever. The coexistence of *openness* and *universality* may seem difficult, but it is implied by the philosophical assumption that there is a certain *invariant* aspect of human thought, that is independent of any historical epoch. I would say that this invariance is a linguistic faculty, which I contend is the condition for memory of *facts*, and therefore a condition for *the world as we know it.* When we encounter completely novel objects, we engage with them via *explication*[1] – that is, we try to provide adequate descriptions that somehow capture the 'nature' of the object. It is this action of explication, which takes place between 'subject' and 'object', that is the condition for the emergence of *phenomena.* I consider this action to be that of *structuring.*

The following dicta serve as an informal basis:

[1]See [5, Ch. 1] for Carnap's account of explication, or also Chapter 6 for my own discussion.

Dictum 3.1. *Being = Thought, and Being/Thought consists of* structures + the act of structuring.[2] *Hence*

$$\textit{structures + the act of structuring = Thought = Being.}$$

Dictum 3.2. *Thought takes place* between *two limits: a* psychological *limit and a* physical *limit. These two limits are often referred to as 'subject' and 'object'. However, subjecthood and objecthood are themselves generated via the relation between the psychological limit and the physical limit.*

Dictum 3.3. *It is the 'betweenness' of the psychological and the physical that* generates *Thought.*

Dictum 3.4. *What 'exists' at the two poles of the psychological and the physical I call* non-Being.

Dictum 3.5. *The pole of the psychological, independent of the physical, is the limit point of certain kinds of spiritual states. In such states, phenomena vanish, as does Thought. It is a state of consciousness without form and without concepts. It is the fact that we have memory of such experiences that enables us to speak of these states* after the states themselves cease. *Hence such experiences become the object of Thought, while the experiences themselves remain external to Thought.*

Dictum 3.6. *No claim can be made about the pole of the physical, since any claim about the physical* necessarily *entails the relation between psychological and physical.*

Dictum 3.7. *A universal method for encoding phenomena thus corresponds to a universal method of Thinking. A universal method of Thinking, in turn, corresponds to a universal ontology, insofar as Being = Thought.*

3.1.1 Composition Theory as an Abstract Hylomorphism

SUMMARY. The connection between Composition Theory and Aristotle's hylomorphic theory of beings is presented. We will see that what we call *compositions* correspond to what Aristotle calls *substances.*

— § —

Aristotle's hylomorphic theory[3] contends that beings consist of matter and form. The matter of an object is the 'stuff' from which it is made, whereas the form is how the matter is organized. For instance, a wooden chair has wood as its matter and the chair shape as its form. Such a compound of matter and form is what Aristotle calls a *substance.* However,

[2]This is in line with the Parmenidean thesis that Thought is Being.
[3]See [2] and [1] in [3].

substances can themselves serve as the matter for a higher order substance. For instance, a wooden chair is a substance, and a classroom can also be conceived as a substance that has for some of its material wooden chairs. Another example is words. The matter of words is letters, whereas the form is the ordering of letters. Words of course serve as the matter of sentences, and sentences serve as the matter for paragraphs, and so on. Thus, the hylomorphic ontology conceives of beings as hierarchical constructs that are recursively defined in terms of matter and form.

Just as substances are built by iterative processes of forming, they can be unpacked by iterative processes of unforming. The latter consists of iteratively extracting the matter from a substance, until eventually a level of 'prime matter' is arrived at. Aristotle calls this primary material 'pure potentiality', and it is characterized by being able to take any form whatsoever.

The Composition Theory that we will present in this chapter realizes an abstract kind of hylomorphism. We are able to define e.g. words as compositions in the same way that Aristotle conceives of words as substances: i.e., they consist of letters for material and have the form of a total order. In the following formalism, we would say that a word is a (compound) composition whose lower level compositions are letters, and which is formed by mapping these letters to totally ordered sets. Likewise we would say that these letters are compositions, and so on.

However, compositions are abstract objects, and are built on the structures defined in Chapter 2. Due to the the abstract nature of compositions, we have no reason to assert that there is an absolute 'prime matter' which is the origin of all things. As we will see, we *do* have primary levels of compositions – called *elementary compositions* – but they are relative to context, and are simply the starting points for higher order constructions. Furthermore, the elementary compositions are defined analogously to compound compositions, and therefore themselves have a form (or, to be more precise, a structure). There is thus no composition that does not have a form.

Our composition theory is also more general than Aristotle's hylomorphic theory. For instance, we can construct a composition that is not so much a *formed material*, but a *concrete realization* of an *abstract structure*. As an example, we can define something very similar to a set-valued functor $F : J \to \mathbf{Set}$ as a composition.[4] In this case, J is like an abstract structure that is given a concrete realization via F. This is a common situation in database management theory, where there is a *schema*, corresponding to J in our example, and a *database*, corresponding to $F(J)$. The functor F is an *instantation* of the schema J; in other words, F is a *concrete realization* of an *abstract structure*.[5]

Our concept of a composition also extends beyond mere *particulars*, and accounts for *classes* as well. A composition corresponding to a particular is called a *determinate composition*, whereas a composition corresponding to a class is called a *determinable composition*.

[4]See Example 3.2 below.

[5]See [6].

This is an important generalization of Aristotelian hylomorphism since in many situations, when we refer to an object we are implicitly referring to a class of objects. For instance, in music if we refer to a note via its pitch-class name, such as C, we are implicitly referring to the *class* of C pitches of all the octaves.

The above discussion provides the intuition for what a composition is. However, since our definition of composition is grounded upon mathematical foundations, it is difficult to generalize informally about the full semiotic range covered by the concept of a composition. Like many situations in mathematics, the generality of the formalism transcends what is comprehensible intuitively. Therefore we are limited in what we can say informally about compositions: at some point, we must simply jump into the deep end and wrestle with the beasts.

3.1.2 Towards a Universal Method of Encoding

Summary. We outline the formal objective of this chapter, which is to achieve universal method of encoding.

— § —

The formal objective of this chapter is to outline a hierarchic construction scheme for compositions, where a composition from each 'level' of composition *encodes* the information of all lower level compositions that constitute it. In particular, this requires the encoding of the *structures* of the lower level compositions. For instance, if we want to define a composition consisting of spatial objects ordered in sequence, then we would want to define an ordering of those objects while still encoding their spatial information. This situation contrasts with material set theories such as ZFC, where we can have sets consisting of structured objects, yet where the structures of those objects is not encoded in the set, since the set is only defined up to its elements. For instance, if we have a set X of topological spaces, we can impose a total order on X to endow it with structure. Nonetheless, the *set* X itself does not encode the topological structures, since the topological spaces are merely *elements* of X. This is not the intuitive situation, since if we observed e.g. a collection of material objects (with topological structure), then the structures of those objects would 'fill' the collection, instead of the structures being replaced by the abstract 'element' relation. An example will make this idea clear.

Suppose we have a collection B of bricks, and we wish to build a house with them by organizing them in space. We can imagine the forming of these bricks into the structure of a house by mapping them into three-dimensional space $\mathbb{R}^3$. The problem arises when we try to map B into $\mathbb{R}^3$, via some function $f : B \to \mathbb{R}^3$. In this case, we can only map a brick $b \in B$ to a single point in $\mathbb{R}^3$. Obviously a brick does not take up a single coordinate in $\mathbb{R}^3$, since the brick itself has a spatial constitution. What we would want to do is define some three-dimensional structure for each brick, and then map these three-dimensional objects into $\mathbb{R}^3$. But we want to define a structure on the set of *all* bricks, in order to construct a *single* composition (viz. a house). So we need a map from the collection of bricks into $\mathbb{R}^3$, but these bricks *cannot be represented as mere points*, as they are in set theory. Therefore we need a 'collection' of these bricks that contains the information of their three-dimensional structure, while still preserving the distinction between each brick. We will see that there is a natural way to do this via colimit structures.

Our second objective is to define a universal scheme for defining higher order compositions in terms of a given set of lower level compositions. This way, each new level of composition is generated via application of the scheme.

Achieving these objectives will enable us to formally define a hierarchy of compositions.

3.2 Elementary Compositions

Summary. Elementary compositions serve as the basis for all higher order compositions. We thus present a formal definition, along with some examples. We then demonstrate how to canonically encode elementary compositions as structures, which is a necessary step for constructing multilevel compositions.

— § —

3.2.1 Formal Definition

Summary. The formal definition of an elementary composition is presented, along with some examples.

— § —

Definition 3.1 (Elementary Composition)**.** An *elementary composition* K is a triple $K = (N, D, C)$, where:

1. N is the *name* of K; it consists of a string of symbols from the free monoid $\mathfrak{N}$ over some alphabet A. The alphabet can be thought to consist of all symbols from all languages, both formal and informal, to allow for maximal freedom when it comes to naming. We refer to the name of K via Name(K), or if no confusion is likely then simply N(K).

2. D is one of the symbols **Determinate** or **Determinable**; it is called the *determination type* of K, and denoted by Determination(K), or if no confusion is likely then simply D(K).

3. C depends on T as follows:

   (a) If $D =$ **Determinate**, then C is a structure morphism $\varphi : S \to S'$.

(b) If $D = \textbf{Determinable}$, then C is a family of structure morphisms $\Phi = \big\{(\varphi_i : S_i \to S_i')\big\}_{i \in I}$.

C is called the *content* of K, and denoted by $\mathrm{Content}(K)$, or if no confusion is likely then simply $\mathrm{C}(K)$. We refer to the domain(s) of C by $\mathrm{dom}(C)$. Thus if D is **Determinate** then $\mathrm{dom}(C)$ is a structure, whereas if D is **Determinable** then $\mathrm{dom}(C)$ is a set of structures. Likewise, we refer to the codomain(s) of C by $\mathrm{cod}(C)$. Furthermore, if D is **Determinable**, then each element $\varphi_i \in \Phi$ is called a *determination* of K.

To denote an elementary composition $K = (N, D, C)$, we write

$$N : D\big(C\big). \tag{3.1}$$

We will see examples later in this section.

We now provide the intuition for each parameter of a structure.

1. *Name.* Compositions are provided with names for semiotic reasons. Since the content of a composition is the 'meat' of the composition, we can imagine that the name refers to the content.

2. *Determination type.* The determination type specifies whether the composition is a *particular* or a *class*. Allowing a composition to be a class is important for situations in which we want to denote an *individual* object which can be realized in diverse ways. For instance, suppose we wish to denote the letter X. In the abstract, we could say that X is a determinate composition; however, according to another view, we can conceive of X as a class of compositions, each of which is a particular realization of X, so that such determinations of X differ with respect to e.g. font. Nonetheless, we still would like to refer to X as a *single* object, which is why we allow for determinable compositions.

3. *Content.* The content is the 'meat' of a composition. In some contexts, it will make sense to think of the domains of the content as abstract structures that are realized concretely by embedding them in the codomains. For instance, if we have the content $\varphi : G \to S$ where G is a directed graph and S some state space, then $\varphi(G)$ realizes the process of moving between states in S. So we have the abstract structure of G that is realized as a concrete process in S.

We now provide some examples of elementary compositions.

Example 3.1. Letters.

In an Aristotelian fashion, we can define letters as elementary compositions. Considering that a letter can be realized in myriad contexts – for instance, in different fonts and locations – we can let its determination type be **Determinable**.

Let us therefore define the letter X, as an example. First, we will define the domain of its content. Let $I = [0,1]$ be the unit interval, considered as a topological space. We demonstrated how to construct topological spaces as elementary structures in Example 2.2, so let τ be the set of generators that endow $[0,1]$ with its usual topological structure. By Equation 2.22, we then get the presheaf $\mathrm{PSh}(\tau) = Fu$. We thus encode I as the structure

$$I \underset{Fu \mapsto @[0,1]}{\longrightarrow} \mathbf{Simple}(@[0,1]). \tag{3.2}$$

Next, define the singleton structure

$$\mathbf{1} \underset{\mathrm{Id}}{\longrightarrow} \mathbf{Simple}(@1). \tag{3.3}$$

Let $m : \mathbf{1} \to I$ be a structure morphism that maps the single element of $\mathbf{1}$ to the midpoint 0.5 of I. We then define $\mathcal{D}$ as the following diagram

$$\begin{tikzcd} \mathbf{1} \arrow[r, "m"] \arrow[d, "m"'] & I \\ I & \end{tikzcd} \tag{3.4}$$

and take the colimit of $\mathcal{D}$ to get a pushout, thus deriving the structure

$$I \cup_m I \underset{\mathrm{Id}}{\longrightarrow} \mathbf{Colimit}(\mathcal{D}). \tag{3.5}$$

The structure $I \cup_m I$ is the topological space consisting of two copies of I that are connected at their midpoint.[6] This provides the topology of the letter X, which will be used as the domain of the content when we define X as a composition.

Since X can exist at any position in space, we can define three-dimensional space as a vector space V conceived as the structure

$$V \underset{Gu \mapsto @\mathbb{R}^3}{\longrightarrow} \mathbf{Simple}(\mathbb{R}^3). \tag{3.6}$$

We can then map $I \cup_i I$ into V in many different ways to generate different instances of X. Therefore we define X as an elementary composition

$$X : \mathbf{Determinable}\big(\{\varphi_j : I \cup_i I \to V\}_{j \in J}\big). \tag{3.7}$$

We can proceed similarly to construct other letters of the English alphabet.

Example 3.2. Functors.

[6] In topology, such a construction is called an *adjunction space.*

We can construct an elementary composition that behaves very much like a functor, such as $D : J \to \mathbf{Set}$ where J is an index category. To create such an elementary composition, we will translate J and $D(J)$ into structures, and D into a structure morphism.

First, we can translate J into a quiver. Let V be a set in bijection with J_0 and E a set in bijection with J_1. We will use V as a set of vertices and E as a set of edges to construct a quiver isomorphic to J. The usual way to define a quiver is how we defined it in Section 2.4.2, i.e., we define maps $s : E \to V$ and $t : E \to V$, where s maps an edge e to its source vertex and t maps e to its target vertex. However, we will proceed in the opposite way, by defining two maps $s : V \to E$ and $t : V \to E$, where s associates a vertex v to the collection of edges whose source is v and t associates v to the collection of edges whose target is v. Such morphisms are clearly not, in general, functional, since a single vertex v can have multiple edges whose source/target is v. Thus, for $\Gamma = \{s, t\}$, we can define the structure

$$Quiv \underset{\mathrm{PSh}(\Gamma)\rightarrowtail @E}{\longrightarrow} \mathbf{Simple}(@E). \tag{3.8}$$

The idea is that the elements $e \in Quiv$ define a category, where the identity arrows identify objects. This is one way to define a category, since we can replace the objects of any category with the identity morphisms, thus providing an 'arrows only' definition of a category.

Now we define $D(J)$ as a structure. For each $X \in D(J)_0$, we want to define the structure corresponding simply to the set X. Our approach is thus the following. First, we define an elementary structure $\mathscr{X} = (X, R_X)$. R_X must contain each functional 1-addressed point $\mathrm{pt}_x : 1 \to X$, where $\mathrm{pt}_x(1) = x$. Such functions specify the elements of X. We will also need to add some other morphisms to R_X: namely, for each morphism $f : Y \to X$ in $D(J)$, we require that $f \in R_X$. We can denote this set by $\mathrm{Hom}_{D(J)}(-, X)$. Therefore $R_X = \{\mathrm{pt}_x\}_{x \in X} \cup \mathrm{Hom}_{D(J)}(-, X)$, whence we derive the structure

$$\mathscr{X} \underset{\mathrm{PSh}(R_X)\rightarrowtail @X}{\longrightarrow} \mathbf{Simple}(@X). \tag{3.9}$$

Now, replacing each $X \in D(J)_0$ with $\mathscr{X} \in \mathbf{Str}_0$ and each morphism $(f : X \to Y) \in D(J)_1$ with the structure morphism $(\mathscr{f} : \mathscr{X} \to \mathscr{Y}) \in \mathbf{Str}_1$, we get a diagram $\mathcal{D}$ in $\mathbf{Str}$. For each morphism $\mathscr{f} : \mathscr{X} \to \mathscr{Y}$ in $\mathcal{D}$, we get the limit structure

$$lim(\mathscr{f}) \underset{\mathrm{Id}}{\longrightarrow} \mathbf{Limit}(\mathscr{X} \xrightarrow{\mathscr{f}} \mathscr{Y}) \tag{3.10}$$

consisting of ordered pairs $(\mathscr{x}, \mathscr{y})$ such that $\mathscr{f}(\mathscr{x}) = \mathscr{y}$. The limit structure thus encodes the structure morphism as a structure. Thus, for all functions $D(J)_1$, we get the corresponding collection of structure morphisms $\mathscr{F}$, and for each $\mathscr{f} \in \mathscr{F}$ we encode it as a limit structure. We then take the coproduct of all such structures

$$coprod\big(lim(\mathscr{f})_{\mathscr{f} \in \mathscr{F}}\big) \underset{\mathrm{Id}}{\longrightarrow} \mathbf{Colimit}(lim(\mathscr{f}), \ldots, lim(\mathscr{h})), \tag{3.11}$$

and use this to obtain the power structure

$$Str(D(J)) \underset{\mathrm{Id}}{\longrightarrow} \mathbf{Power}\big(coprod\big(lim(\mathscr{f})_{\mathscr{f}\in\mathscr{F}}\big)\big), \tag{3.12}$$

which corresponds to $D(J)$. $Str(D(J))$ contains all subobjects of $coprod\big(lim(\mathscr{f})_{\mathscr{f}\in\mathscr{F}}\big)$, and thus contains $lim(\mathscr{f}), \ldots, lim(\mathscr{h})$.

Now we can define a structure morphism $\mathscr{D} : Quiv \to Str(D(J))$ that corresponds to $D : J \to \mathbf{Set}$. $\mathscr{D}$ sends each edge $e \in Quiv$ to $\mathscr{D}(e) = lim(\mathscr{f}) \in Str(D(J))$. Specifically, for $\alpha : i \to j$ a morphism in J such that $D(\alpha) = f : X \to Y$ in $\mathbf{Set}$, we get an edge $e_\alpha \in Quiv$ such that

$$\mathscr{D}(e_\alpha) = lim(\mathscr{f}). \tag{3.13}$$

Note that for an object i in J such that $J(i) = X$, we get an edge $e_{\mathrm{Id}_i} \in Quiv$ such that

$$\mathscr{D}(e_{\mathrm{Id}_i}) = lim(\mathrm{Id}_{\mathscr{X}}). \tag{3.14}$$

However, since $lim(\mathrm{Id}_{\mathscr{X}}) \simeq \mathscr{X}$, we can just let $\mathscr{D}(e_{\mathrm{Id}_i}) = \mathscr{X}$.

Thus we encode a functor $D : J \to \mathbf{Set}$ as an elementary composition

$$DComp : \mathbf{Determinate}\big(Quiv \xrightarrow{\mathscr{D}} Str(D(J))\big). \tag{3.15}$$

Let's see a concrete example. Let $Quiv$ denote the set of edges from the following index category J

$$\begin{tikzcd} i \arrow[dr, "\alpha"] & \\ & j \arrow[dl, "\beta"] \\ k \arrow[uu, "\gamma"] & \end{tikzcd} \tag{3.16}$$

and suppose we have $D(J)$ as the following diagram in $\mathbf{Set}$:

$$\begin{tikzcd} D(i) = X \arrow[dr, "D(\alpha)=f"] & \\ & D(j) = Y \arrow[dl, "D(\beta)=g"] \\ D(k) = Z \arrow[uu, "D(\gamma)=h"] & \end{tikzcd} \tag{3.17}$$

Suppose $X = \{a, b\}, Y = \{p, q, r\}, Z = \{x, y\}$, and let f, g, h be given by

$$\begin{tikzcd}
\underline{X} & \underline{Y} & \underline{Z} & \underline{X} \\
a \arrow[r, "f"] & p \arrow[r, "g"] & x \arrow[r, "h"] & a \\
b \arrow[r, "f"] & q \arrow[ru, "g"] & y \arrow[r, "h"] & b \\
 & r \arrow[ru, "g"] & &
\end{tikzcd} \tag{3.18}$$

We translate X, Y, Z into $\mathcal{X}, \mathcal{Y}, \mathcal{Z}$, and thus we get the limit structures

$$\begin{aligned}
&(1) \quad lim(\mathscr{f}) \underset{\text{Id}}{\longrightarrow} \mathbf{Limit}(\mathcal{X} \xrightarrow{\mathscr{f}} \mathcal{Y}) &&\rightsquigarrow \quad \{(a,p),(b,q)\},\\
&(2) \quad lim(\mathscr{g}) \underset{\text{Id}}{\longrightarrow} \mathbf{Limit}(\mathcal{Y} \xrightarrow{\mathscr{g}} \mathcal{Z}) &&\rightsquigarrow \quad \{(p,x),(q,x),(r,y)\},\\
&(3) \quad lim(\mathscr{h}) \underset{\text{Id}}{\longrightarrow} \mathbf{Limit}(\mathcal{Z} \xrightarrow{\mathscr{h}} \mathcal{X}) &&\rightsquigarrow \quad \{(x,a),(y,b)\},
\end{aligned} \tag{3.19}$$

which correspond to the set maps f, g, h. The squiggly arrows pointing to the sets of pairs shows what the limit structures correspond to as sets.

Taking the coproduct of the structures, we get

$$coprod\big(lim(\mathscr{f})_{\mathscr{f} \in \mathcal{F}}\big) \underset{\text{Id}}{\longrightarrow} \mathbf{Colimit}(\mathcal{X}, lim(\mathscr{f}), \mathcal{Y}, lim(\mathscr{g}), \mathcal{Z}, lim(\mathscr{h})) \tag{3.20}$$

and then, as in Equation 3.12, the power structure

$$Str(D(J)) \underset{\text{Id}}{\longrightarrow} \mathbf{Power}(coprod\big(lim(\mathscr{f})_{\mathscr{f} \in \mathcal{F}}\big)),$$

which contains $\mathcal{X}, lim(\mathscr{f}), \mathcal{Y}, lim(\mathscr{g}), \mathcal{Z}, lim(\mathscr{h})$ as elements.

Thus we have

$$DComp : \mathbf{Determinate}\big(Quiv \xrightarrow{\mathcal{D}} Str(D(J))\big)$$

as the elementary composition whose content is depicted as in Figure 3.1. The idea is that for each morphism $\alpha : i \to j$ in J that gives $D(\alpha) = f : X \to Y$ in **Set**, we have α represented as an edge in $Quiv$, and f represented as a limit structure $lim(\mathscr{f})$ in $Str(D(J))$. Although an unusual encoding, it does indeed get the job done of translating a functor into an elementary composition.

Example 3.3. Points of a structure.

As a final example, we consider the situation in which one wishes simply to pick out a point of a structure. For instance, suppose one wanted to specify a composition as some

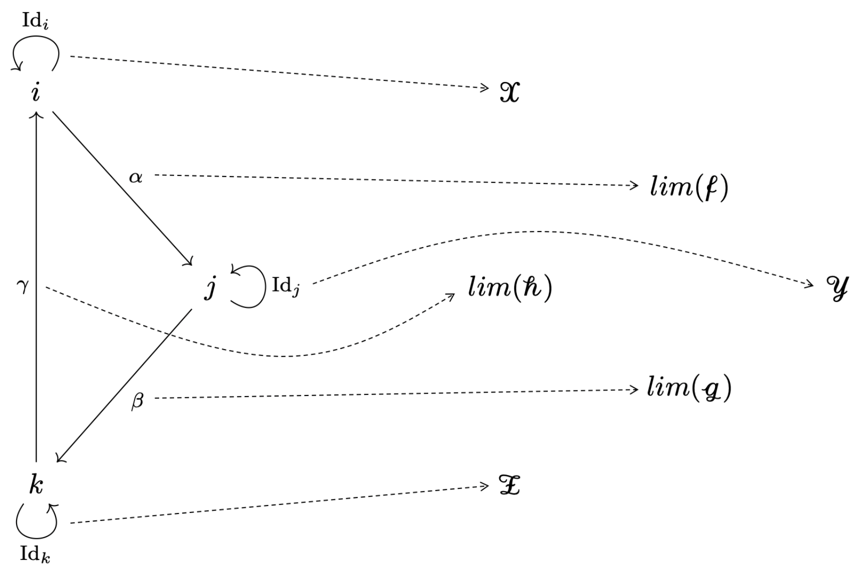


Figure 3.1: To the left is the quiver $Quiv$ and to the right the structure $Str(D(J))$. Note that the edges in $Quiv$ correspond to limit structures in $Str(D(J))$, because the limit structures encode the morphisms that are expressed by the original diagram $D(J)$.

collection of frequencies, in order to specify a sonic spectrum. In this case, we can define a structure

$$Freq \underset{Fu\mapsto @\mathbb{R}}{\longrightarrow} \mathbf{Simple}(\mathbb{R}) \tag{3.21}$$

that defines a vector space on $\mathbb{R}$. Then we can take

$$FreqColls \underset{\mathrm{Id}}{\longrightarrow} \mathbf{Power}(Freq) \tag{3.22}$$

to get the structure that contains of all subsets of frequencies. Then we can simply define a functional structure morphism $f : \mathbf{1} \to FreqColls$ to specify a set of frequencies, thus providing our elementary composition

$$thisSpectrum : \mathbf{Determinate}\big(\mathbf{1} \xrightarrow{f} FreqColls\big). \tag{3.23}$$

For such a composition, there is also another way to denote the same collection of frequencies from $FreqColls$. This involves Mazzola's concept of a denotator as discussed in [15], in the context of the category **Mod** of modules. For modules M, N, and $@N$ the representable functor of N, an M-addressed denotator corresponds to an element $d \in M@N$, where $M@N$ is the set of morphisms from M to N in **Mod**. We say that d is an element of the functor $@N$.

The idea, then, is that we can denote a subset of frequencies in $Freq$ as an element of the functor of $FreqColls$. For instance, by Convention 2.1, we can assume that the functor Fu of $Freq$ contains all functional 1-addressed points of $\mathbb{R}$. Hence, for every $x \in \mathbb{R}$, we have an element $\mathrm{pt}_x \in 1@Fu$ that picks out x. Thus there will be a subfunctor $Gu \rightarrowtail Fu$ such that $1@Gu \subset 1@Fu$, where $1@Gu$ determines a subset $A \subset \mathbb{R}$. Therefore when we take the power structure $FreqColls$, we get the set $1@Gu$ corresponding to an element $A \in 1@\Omega^{Fu} \simeq 1@FreqColls$. Thus we can say that the subset A of frequencies corresponds to a 1-addressed denotator of $FreqColls$.

Nonetheless, since we can encode denotators as elementary compositions, it is not necessary to incorporate the concept of a denotator.

3.2.2 The Canonical Encoding of Elementary Compositions as Structures

Summary. The canonical method of translating elementary compositions into structures is presented. This is a crucial step for defining higher order compositions.

— § —

We can translate elementary compositions into structures in a canonical way. The method of translation depends on whether a composition is determinate or determinable. The former case is simple while the latter requires some more explanation, so we'll start with the former.

The reason that we discuss these translations is practical, since it is a necessary step in defining compound compositions. We will see later that once we define a collection of compositions, we translate them into structures, and use these structures as the material of higher order compositions. There is thus a recursive scheme for defining higher order compositions in terms of lower level compositions.

3.2.2.1 Translation of a Determinate Composition Into a Structure

Summary. We provide the canonical method for translating determinate compositions into structures.

— § —

Let

$$Comp : \mathbf{Determinate}\big(S \xrightarrow{\varphi} T\big)$$

be a composition. We translate it into a structure via

$$CompAsStructure \xrightarrow[\mathrm{Id}]{} \mathbf{Limit}(S \xrightarrow{\varphi} T). \tag{3.24}$$

So we are translating determinate compositions into structures by taking the content (i.e., the structure morphism) of the composition, and using it as the coordinator of a limit structure. Thus, we are essentially mapping structure morphisms to limit structures, which we may define as the map

$$\mathfrak{Str} : \mathfrak{N} \times \mathbf{Str}_1^{@} \to \mathbf{Str}_0^{@} : (\varphi : S \to T) \mapsto \big(N \underset{\mathrm{Id}}{\longrightarrow} \mathbf{Limit}(S \xrightarrow{\varphi} T)\big). \tag{3.25}$$

Note that $\mathfrak{N}$ consists of the namespace used for constructing the names of structures, so therefore

$$\mathfrak{Str}(N, \varphi) = N \underset{\mathrm{Id}}{\longrightarrow} \mathbf{Limit}(S \xrightarrow{\varphi} T). \tag{3.26}$$

3.2.2.2 Translation of a Determinable Composition Into a Structure

Summary. We provide the canonical method for translating determinable compositions into structures.

— § —

Let

$$\mathit{Comp} : \mathbf{Determinable}\big(\{S_k \xrightarrow{\varphi_k} T_k\}_{k\in K}\big)$$

be a determinable composition. By Equation 3.25, we can encode each of its determinations $\varphi_k : S_k \to T_k$ as a limit structure

$$\mathfrak{Str}(\mathit{Det_kAsStructure}, \varphi_k) = \mathit{Det_kAsStructure} \underset{\mathrm{Id}}{\longrightarrow} \mathbf{Limit}(S_k \xrightarrow{\varphi_k} T_k).$$

Let $\mathbf{Str}(\mathit{Comp})$ denote the subcategory of $\mathbf{Str}$ consisting of the following:

1. The set $\mathbf{Str}(\mathit{Comp})_0$ of objects consists of all determinations $\mathit{Det_kAsStructure}$ of Comp, taken as structures.
2. The set $\mathbf{Str}(\mathit{Comp})_1$ of morphisms consists of all structure morphisms that exist between the objects in $\mathbf{Str}(\mathit{Comp})_0$.

Now, define a diagram $\mathcal{D} : J \to \mathbf{Str}(\mathit{Comp})$ that is bijective on objects. We then encode Comp as a structure via

$$\mathit{CompAsStructure} \underset{\mathrm{Id}}{\longrightarrow} \mathbf{Colimit}(\mathcal{D}).$$

A determination of $\mathit{CompAsStructure}$ can also be encoded as a structure. For each determination φ_k, we get the canonical injection $\mathit{Det}_k \overset{\iota_k}{\hookrightarrow} \mathit{CompAsStructure}$. This thus gives back the structure

$$\mathit{Det_kAsStructure} \underset{\mathrm{Id}}{\longrightarrow} \mathbf{Limit}(\mathit{Det}_k \overset{\iota_k}{\hookrightarrow} \mathit{CompAsStructure}).$$

The reason that we defined the diagram $\mathcal{D}$ is because it has the potential to determine an equivalence relation on the determinations of *CompAsStructure*. To see why this is important, consider Example 3.1 where we discussed the letter X as a determinable composition. Translating X into a structure, we'd have

$$XAsStructure \underset{\mathrm{Id}}{\longrightarrow} \mathbf{Colimit}(\mathcal{D}).$$

In this case, we have the encoding of each determination φ_k of X as a structure

$$XDet_k \underset{\mathrm{Id}}{\longrightarrow} \mathbf{Limit}(I \cup_m I \xrightarrow{\varphi_k} V).$$

Recall that K is the index set of the determinations $\{\varphi_k\}_{k \in K}$. For each $j, k \in K$, it makes sense to define an epimonic map $\zeta : XDet_j \to XDet_k$ such that the diagram

$$\begin{tikzcd} XDet_j \arrow[r, "\zeta"] \arrow[d, "\mathrm{pr}_1"'] & XDet_k \arrow[d, "\mathrm{pr}_1"] \\ I \cup_m I \arrow[r, "\mathrm{Id}"'] & I \cup_m I \end{tikzcd} \tag{3.27}$$

commutes. The commutativity of the diagram means that ζ's action on the first coordinate $I \cup_m I$ of $XDet_j$ corresponds to the identity map $\mathrm{Id}_{I \cup_m I}$. So we can conceive of ζ as a pair $\zeta = (\mathrm{Id}_{I \cup_m I}, f)$, where $f : V \to V$. This is because $I \cup_m I$ is fixed for all determinations of X, and the only thing that changes between X's determinations is the spatial incarnation in V of the topology of $I \cup_m I$. Therefore it makes sense to define epimonisms between all determinations $XDet_j$ of $XAsStructure$, since this establishes the equivalence between any two points $(i, v) \in XDet_j$ and $(i', v') \in XDet_k$ under the condition that $i = i'$. We will see later why establishing such equivalences is important.

3.3 Steps to Constructing Composition Hierarchies

Summary. We take the necessary steps in defining hierarchies of compositions. We first present the notion of a *basis of compositions*, which is a basis from which we construct higher order compositions, called *compound compositions*. However, since it is not so straightforward to define compound compositions, we must take an intermediate step and define the notion of a *composition constructor*. This will enable us to easily define what is called a *super domain*. The way that all of these concepts hang together naturally leads us to define what constitutes a *hierarchy of compositions*, which is a crucial notion that guarantees the universality of our ontology.

— § —

Like Aristotle, our main idea is that compositions are hierarchical objects that are composed of lower level compositions, and that these lower level compositions are, in turn, composed of even lower level compositions, and so on. Thus, for a set $\mathbf{B}$ of compositions, we can construct new compositions using those of $\mathbf{B}$ as material. We call $\mathbf{B}$ a *basis of compositions*, and a set Σ of new compositions composed with those of $\mathbf{B}$ a *super domain of compositions*. We will see, however, that we cannot conceive of such sets as actual sets, because of the discussion in Section 3.1.2. Therefore we will have to convert sets of compositions into another type of object that is not a set, but which behaves very much like a set.

3.3.1 Basis of Compositions

SUMMARY. We present the notion of a *basis of compositions*, which will require us to translate compositions into structures. We will see that a basis of compositions corresponds to a diagram in **Str**.

— § —

A *basis of compositions* $\mathbf{B}$ is a collection of compositions used to construct higher order compositions. For instance, with respect to words, we can say that a set $\mathbf{B}$ of letters is a basis of compositions.

If $\mathbf{B}$ is a set of elementary compositions, then $\mathbf{B}$ is called an *absolute* basis of composition, since there are no lower level compositions from which each $K \in \mathbf{B}$ is constructed. For any basis of compositions $\mathbf{B}$, a collection of compositions constructed with the compositions of $\mathbf{B}$ is called a *super domain of compositions (over* $\mathbf{B}$*)*. When the context is clear, we will simply refer to a basis of compositions as a *basis*, and a super domain of compositions as a *super domain*.

Given a basis $\mathbf{B}$, we need to translate each composition $K \in \mathbf{B}$ into a structure. We require this translation because in order to construct higher order compositions, we need to combine compositions in $\mathbf{B}$ together, and then convert this combination into a structure to be used as the domain of the content of a new composition. The idea is that we are *forming* the compositions into a higher order object. This is like a composer constructing musical figures with a collection of pitches, or someone building a house with a collection of bricks.

Our current task is to translate $\mathbf{B}$ into a category whose objects are structures. Specifically, we will define a functor $\mathfrak{B} : \mathbf{B} \to \mathbf{Str}$, where $\mathbf{B}$ is a discrete category.[7] Our translations of determinate and determinable compositions in Section 3.2.2 laid the groundwork for performing this translation, so let us now turn to how we define the functor $\mathfrak{B}$.

For a composition K, to be concise let $\mathrm{dom}(K)$ resp. $\mathrm{cod}(K)$ denote $\mathrm{dom}(\mathrm{Content}(K))$ resp. $\mathrm{cod}(\mathrm{Content}(K))$. Given $\mathbf{B}$, we define $\mathfrak{B} : \mathbf{B} \to \mathbf{Str}$ as follows. For each $K \in \mathbf{B}$:

[7] That is, we interpret $\mathbf{B}$ as a category whose objects are the elements $K \in \mathbf{B}$ and where we only have identity morphisms.

1. If K is **Determinate**, then $\mathfrak{B}$ sends K to the limit structure

$$Str(K) \underset{\mathrm{Id}}{\longrightarrow} \mathbf{Limit}(\mathrm{dom}(K) \xrightarrow{\varphi} \mathrm{cod}(K)).$$

2. If K is **Determinable**, then:

   (a) Translate each determination φ_i of K into the limit structure

   $$Str(K_{Det_i}) \underset{\mathrm{Id}}{\longrightarrow} \mathbf{Limit}(\mathrm{dom}(K)_i \xrightarrow{\varphi_i} \mathrm{cod}(K)_i).$$

   Let $\mathbf{Str}(K_{Dets})$ denote the subcategory of $\mathbf{Str}$ that contains all determinations $Str(K_{Det_i})$ as objects and all structure morphisms $\eta : Str(K_{Det_i}) \to Str(K_{Det_j})$ between any two such structures as morphisms.

   (b) Construct a diagram $\mathcal{D} : J \to \mathbf{Str}(K_{Dets})$ that is bijective on objects.

   (c) Derive the colimit structure

   $$Str(K) \underset{\mathrm{Id}}{\longrightarrow} \mathbf{Colimit}(\mathcal{D}).$$

   Thus $\mathfrak{B}$ sends K to $Str(K)$.

We can specify $\mathfrak{B}$ as a pair $\mathfrak{B} = (\mathbf{B}, Dia)$, where:

1. $\mathbf{B}$ is a basis.

2. $Dia : \mathbf{B}_{|C} \to \mathscr{D}$ is a map from the subset $C \subseteq \mathbf{B}$ of determinable compositions in $\mathbf{B}$ to the set[8] of all diagrams in $\mathbf{Str}$, sending each $K \in C$ to a diagram $\mathcal{D} : J \to \mathbf{Str}(K_{Dets})$, as outlined above.

This would provide the necessary data to define the functor $\mathfrak{B} : \mathbf{B} \to \mathbf{Str}$.

Since $\mathbf{B}$ is a discrete category, then $\mathfrak{B}(\mathbf{B})$ is likewise discrete. In the next section, we will see how to use $\mathfrak{B}$ to define constructions of higher order compositions. From now on, we will refer to $\mathfrak{B}(\mathbf{B})$ as simply $\mathfrak{B}$.[9]

3.3.2 Compound Compositions

Summary. Compound compositions are compositions that are built from lower level compositions. Although this idea is simple in its conception, to define compound compositions rigorously is rather difficult. We therefore must provide some technical groundwork.

[8]To avoid issues of this set being large, we can restrict to the small set of diagrams $\mathcal{D} : J \to \coprod_{K \in \mathbf{B}} \mathbf{Str}(K_{Dets})$.

[9]Note that this is simply the usual way we denote diagrams. That is, for a diagram $\mathcal{D} : J \to \mathscr{C}$, we don't refer to $\mathcal{D}(J)$, but simply $\mathcal{D}$ for short.

— § —

When we define an absolute basis $\mathbf{B}$ of elementary compositions, we can use these compositions to create new compositions. We call these higher order compositions *compound compositions.*

3.3.2.1 Composition Accumulators

SUMMARY. Composition accumulators correspond to multisets of compositions, and are to be used as the material for higher order compositions. However, we cannot define composition accumulators simply as multisets, so we must construct a new kind of object that behaves like a multiset, yet which is distinct.

— § —

Given a basis $\mathbf{B}$, a compound composition S over $\mathbf{B}$ consists of a collection of compositions from $\mathbf{B}$ that are formed together in some way. For an elementary composition $K \in \mathbf{B}$, we allow multiple occurrences of K in S. For example, if we have $\mathbf{B}$ consisting of atoms from the periodic table and we wish to construct an H_2O molecule as a compound composition, then H_2O consists of 2 hydrogen atoms and 1 oxygen atom. For a compound composition S, we call its subset $X \subseteq \mathbf{B}$ of basis compositions the *support* of S and the function assigning each $S \in X$ to its quantity of occurrences in S the *multiplicity* of S. Thus for H_2O, its support is the set $X = \{\text{hydrogen, oxygen}\}$ and its multiplicity is the function $m : X \to \mathbb{N}$ such that $m(\text{hydrogen}) = 2$ and $m(\text{oxygen}) = 1$. Therefore the pair (X, m) determines a multiset. We call such a pair $\mathcal{A} = (X, m)$ a *composition accumulator* over $\mathbf{B}$.

Since we are using the category $\mathfrak{B}$ to carry out compound composition constructions, we want to translate the above multiset convention into a categorical setting. Let J be a discrete index category. We get a *composition accumulator* $\mathcal{A}$ over $\mathbf{B}$ as the colimit of a diagram $\mathcal{D} : J \to \mathfrak{B}$. Since J and $\mathfrak{B}$ are discrete, $\mathcal{D}$ is in essence a set function, and $\mathcal{A} = colim(\mathcal{D})$ is a coproduct of structures. $\mathcal{D}(J)$ gives the support of $\mathcal{A}$, denoted by $\mathfrak{sup}(\mathcal{A})$, whereas the fiber of each element $K \in \mathcal{D}(J)$ gives the multiplicity function of $\mathcal{A}$, denoted by $\mathfrak{mul}(\mathcal{A})$. For example, if $J = \{j_1, j_2\}$ and $\mathcal{D}(j_1) = \mathcal{D}(j_2) = K$, then $\mathfrak{sup}(\mathcal{A}) = \{K\}$ and $\mathfrak{mul}(\mathcal{A})(K) = 2$. This means that $\mathcal{A}$ *accumulates* two instances of the composition $K \in \mathbf{B}$ (or, equivalently, two instances of the structure $Str(K) \in \mathfrak{B}$).

Note that in the naive setting, we defined $\mathcal{A}$ as a multiset, where X was the support and m assigned multiplicities to the elements $x \in X$. In the categorical setting, however, $\mathcal{A}$ is *not* a multiset, but a disjoint union of structures. We need to explain why this is important. It is clear that each $K \in \mathbf{B}$ contains deep information, such as whether it is determinate or determinable, and, especially, what its content is. But if we had a multiset $\mathcal{A}$ over $\mathbf{B}$, we could not *encode* this information of each K within $\mathcal{A}$, since an element in a multiset *is only defined insofar as it stands in the element relation '$\in$' to* $\mathcal{A}$. In other words, each $K \in \mathbf{B}$ would be reduced to a mere element, and thus we would be eradicating the 'essence' of K once we say that $K \in \mathcal{A}$, precisely because K is thereby reduced to nothing more than the fact that it stands in the element relation to $\mathcal{A}$.

On the other hand, defining $\mathcal{A}$ as the colimit $\mathcal{A} = colim(\mathcal{D})$ preserves the depth of K in $\mathcal{A}$, while nonetheless preserving the distinction between each K, since $\mathcal{A}$ is a disjoint union. Thus, instead of the informationally poor element relation $K \in \mathcal{A}$, which totally conceals the nature of K, we get the informationally rich inclusion relation $K \hookrightarrow \mathcal{A}$, which includes all of the nature of K.[10] This is an absolutely crucial aspect for defining composition hierarchies, which is why we encapsulate it in the following

Dictum 3.8. *The canonical injective relation $K \hookrightarrow \mathcal{A}$ that holds between a basis composition K and a composition accumulator $\mathcal{A}$ is what enables any level of a composition hierarchy to encode all information about compositions lower in the hierarchy.*

This proposition means that ascent to higher levels of composition is not merely abstract, but that each ascensional step preserves all of the lower level steps in a concrete and transparent way.

3.3.2.2 Realizing Compound Compositions

Summary. The technical machinery for realizing compound compositions is provided. We also provide an example of the significance of the equivalence relation discussed in Section 3.2.2.

— § —

A composition accumulator $\mathcal{A}$ corresponds to a colimit structure. If each basis composition $K \hookrightarrow \mathcal{A}$ is determinate, then $\mathcal{A}$ will be used as the domain of the content of a compound composition. The idea is that the compositions in $\mathcal{A}$ serve as the material, and a structure morphism $\psi : \mathcal{A} \to F$ forms (i.e., structures) the material in some way. However, if any basis compositions $K \hookrightarrow \mathcal{A}$ are determinable, then there are more steps to take care of.

We will now consider the four main cases of compound composition construction. They are divided into two kinds of cases:

1. $\mathcal{A}$ is either determinate or determinable.
2. The compound composition S of which $\mathcal{A}$ is the material is either determinate or determinable.

More specifically, we have the following cases to consider:

[10] Nonetheless, there are good examples of when it is fruitful to consider $\mathcal{A}$ as a multiset in the 'naive' sense, since it simplifies things. This is an approach we will take in Chapter 4.

1. Each basis composition $K \hookrightarrow \mathcal{A}$ is determinate, and the compound composition S of which $\mathcal{A} = \mathrm{dom}(S)$ is determinate.
2. Each basis composition $K \hookrightarrow \mathcal{A}$ is determinate, whereas the compound composition S of which $\mathcal{A} = \mathrm{dom}(S)$ is determinable.
3. There are basis compositions $K \hookrightarrow \mathcal{A}$ that are determinable, and the compound composition S of which some determination $\mathcal{A}' = \mathrm{dom}(S)$ of $\mathcal{A}$ is determinate.
4. There are basis compositions $K \hookrightarrow \mathcal{A}$ that are determinable, and the compound composition S is determinable, meaning either of the following:
   (a) There are multiple determinations $\mathcal{A}_1, \dots, \mathcal{A}_n$ of $\mathcal{A}$ in $\mathrm{dom}(S)$, while $\mathrm{cod}(S)$ is a singleton, meaning that the domain is fixed over all determinations of S.
   (b) There is only a single determination $\mathcal{A}'$ of $\mathcal{A}$ in $\mathrm{dom}(S)$, while there are multiple codomains in $\mathrm{cod}(S)$.
   (c) A combination of the latter two, where there are multiple determinations $\mathcal{A}_1, \dots, \mathcal{A}_n$ of $\mathcal{A}$ in $\mathrm{dom}(S)$, as well as multiple codomains $F_1, \dots, F_m$ in $\mathrm{cod}(S)$.

We now proceed in tackling each of these situations.

Case 1. Let $\mathcal{D} : J \to \mathfrak{B}$ be a diagram and therefore $\mathcal{A} = colim(\mathcal{D})$ a composition accumulator. If each $K \in \mathcal{D}(J)$ corresponds to a determinate composition, then $\mathcal{A}$ corresponds to a determinate composition. Therefore we define the compound composition

$$Comp : \mathbf{Determinate}\big(\mathcal{A} \xrightarrow{\psi} F\big). \tag{3.28}$$

Note that we could have defined $Comp$ in an equivalent way, which is conceptually distinct. We could have fixed the structure F, and for each $j \in J$, defined a structure morphism

$$\rho_j : \mathcal{D}(j) \to F.$$

This would specify the structure of each $\mathcal{D}(j)$ component of $\mathcal{A}$ individually. We could then 'glue' together all ρ_j morphisms, thus getting the structure morphism

$$Glue(\{\rho_j\}_{j\in J}) : \coprod_{j\in J} \mathcal{D}(j) \to F, \tag{3.29}$$

where $Glue(\{\rho_j\}_{j\in J})$ combines all ρ_j into a single morphism acting on the disjoint union of the domains of all ρ_j for $j \in J$.

Case 2. This is like Case 1, except we have a family of maps $\{\psi_i : \mathcal{A} \to F_i\}_{i\in I}$ instead of a single map $\psi : \mathcal{A} \to F$. Thus we get the following determinable compound composition

$$Comp : \mathbf{Determinable}\big(\{\psi_i : \mathcal{A} \to F_i\}_{i\in I}\big) \tag{3.30}$$

Case 3. In this situation, for a diagram $\mathcal{D} : J \to \mathfrak{B}$, there are some $j \in J$ such that $\mathcal{D}(j) = K$ is determinable. Nonetheless, the compound composition we wish to construct with $\mathcal{A} = colim(\mathcal{D})$ is determinate, so we require a determination of each determinable composition $K \hookrightarrow \mathcal{A}$, thus deriving some new $\mathcal{A}'$ which will be used as the domain of the compound composition.

Recall from Section 3.2.2.2 that if $K \in \mathcal{D}(J)$ is determinable, then we have a determination of K encoded as the limit structure

$$Str(K_{Det_i}) \underset{\mathrm{Id}}{\longrightarrow} \mathbf{Limit}(\mathrm{dom}(K)_i \xrightarrow{\psi_i} \mathrm{cod}(K)_i).$$

For simplicity, we can denote $Str(K_{Det_i})$ simply by K_{Det_i}. Therefore for each $K \in \mathcal{D}(J)$ such that K is determinable, select a determination $K_{Det_i} \hookrightarrow K$. After replacing each determinable composition $K \in \mathcal{D}(J)$ with one of its determinations, we get $\mathbf{Str}(Dets)$ as the category of the resulting determinate compositions (which includes those compositions $L \in \mathcal{D}(J)$ that were already determinate). Thus we can derive the new diagram $\mathcal{D}_{Dets} : J \to \mathbf{Str}(Dets)$, where if $\mathcal{D}(j) = K$ is determinable then $\mathcal{D}_{Dets}(j)$ equals some determination K_{Det_i}. Then we take our new composition accumulator to be $\mathcal{A}' = colim(\mathcal{D}_{Dets})$, and derive therefore the compound composition

$$Comp : \mathbf{Determinate}\big(\mathcal{A}' \xrightarrow{\psi'} F\big). \tag{3.31}$$

Equivalently, like in Case 1, we can define $\rho_j : \mathcal{D}_{Dets}(j) \to F$ for each $j \in J$, and then glue all ρ_j together, deriving

$$Glue(\{\rho_j\}_{j\in J}) : \coprod_{j\in J} \mathcal{D}_{Dets}(j) \to F. \tag{3.32}$$

Case 4. This is like Case 3, except the compound composition is determinable. We can achieve the construction of a determinable compound composition whose support contains determinable compositions in one of the following ways:

1. We can fix a diagram $\mathcal{D}_{Dets} : J \to \mathbf{Str}(Dets)$, getting the composition accumulator $\mathcal{A} = colim(\mathcal{D}_{Dets})$. Then we define the compound composition

   $$Comp : \mathbf{Determinable}\big(\{\mathcal{A} \xrightarrow{\eta_i} F_i\}_{i\in I}\big). \tag{3.33}$$

   In this situation, the domain is fixed over all determinations, while the codomain varies.

2. We can do the opposite, i.e., specify a family of diagrams $\{\mathcal{D}_{Dets_i} : J \to \mathbf{Str}(Dets_i)\}_{i\in I}$ as the domains and specify a fixed codomain F, as so

   $$Comp : \mathbf{Determinable}\big(\{colim(\mathcal{D}_{Dets_i}) \xrightarrow{\psi_i} F\}_{i\in I}\big). \tag{3.34}$$

3. We can do both, i.e., specify a family of diagrams and codomains, thus deriving

$$Comp : \mathbf{Determinable}\big(\{colim(\mathcal{D}_{Dets_i}) \xrightarrow{\psi_i} F_i\}_{i\in I}\big). \tag{3.35}$$

Figure 3.2 provides an intuitive representation for how compound compositions are constructed.

Before moving on to the next section, now is an important time to discuss a topic that was touched upon in Section 3.2.2. There, we required that for a determinable composition K, we encode it as a structure not simply by taking the coproduct of its determinations, but by taking a general colimit potentially containing morphisms between determinations. We saw that this generated an equivalence relation $\sim$. We will now see how this equivalence relation is important.

Let K be a determinable composition (as a structure) and let K_{Det_1} and K_{Det_2} be determinations of K. Let D be a diagram of which K is the colimit, i.e., $K = colim(D)$. Suppose that in D there is an epimonism $f : K_{Det_1} \to K_{Det_2}$. This means that K_{Det_1} and K_{Det_2} are basically equivalent, since f is epimonic and for each $x \in K_{Det_1}$, we have that $x \sim f(x)$.

Now suppose we have two diagrams $\mathcal{D}_1 : J \to \mathbf{Str}(Dets_1)$ and $\mathcal{D}_2 : J \to \mathbf{Str}(Dets_2)$ where they are both equal except that $\mathcal{D}_1$ sends some j to K_{Det_1} whereas $\mathcal{D}_2$ sends j to K_{Det_2}. Then it is also clear that $\mathcal{A}_1 = colim(\mathcal{D}_1)$ should be equivalent to $\mathcal{A}_2 = colim(\mathcal{D}_2)$.

Therefore, if $\mathcal{A}_1$ and $\mathcal{A}_2$ are equivalent, then for structure morphisms $\varphi_1 : \mathcal{A}_1 \to F$ and $\varphi_2 : \mathcal{A}_2 \to F$, if for every $x \in \mathcal{A}_1$, $y \in \mathcal{A}_2$ such that $x \sim y$ we also have that $\varphi_1(x) = \varphi_2(y)$, then φ_1 and φ_2 should also be equivalent. We can see this equivalence by the commutativity of the following diagram

$$\begin{tikzcd}
\mathcal{A}_1 \arrow[rr, "f^*"] \arrow[dr, "\varphi_1"'] & \arrow[d, phantom, "\sim"] & \mathcal{A}_2 \arrow[dl, "\varphi_2"] \\
& F &
\end{tikzcd} \tag{3.36}$$

where $f^* : \mathcal{A}_1 \to \mathcal{A}_2$ is the epimonism induced by $f : K_{Det_1} \to K_{Det_2}$:

Now let

$$Comp : \mathbf{Determinable}\big(\{\mathcal{A}_i \xrightarrow{\varphi_i} F\}_{i\in I}\big)$$

be a determinable composition. Then we have the equivalence relation $\sim$ on Content($Comp$) whereby $\varphi_i \sim \varphi_j$ if, for an epimonic morphism $f^* : \mathcal{A}_i \to \mathcal{A}_j$, the diagram

$$\begin{tikzcd}
\mathcal{A}_i \arrow[rr, "f^*"] \arrow[dr, "\varphi_i"'] & & \mathcal{A}_j \arrow[dl, "\varphi_j"] \\
& F &
\end{tikzcd} \tag{3.37}$$

$$\begin{tikzcd}
 & colim \arrow[dl, dash, dashed] \arrow[dr, dash, dashed] \arrow[rr, "\varphi"] & & F \\
K & \dots & K' &
\end{tikzcd}$$

(a) A determinate compound composition. We take the colimit *colim* of $K, \dots, K'$. Then we complete the construction of the composition by mapping *colim* into F via φ.

$$\begin{tikzcd}
 & colim \arrow[dl, dash, dashed] \arrow[dr, dash, dashed] \arrow[rrr, leftrightarrow] & & & colim_{Det_i} \arrow[dl, dash, dashed] \arrow[dr, dash, dashed] \arrow[rr, "\varphi_i"] & & F_i \\
K & \dots & K' & K_{Det_i} & \dots & K'_{Det_i} &
\end{tikzcd}$$

(b) A determinable compound composition. For the general colimit $colim(K, \dots, K')$, we get a determination as a colimit $colim_{Det_i}$ of the determinations $K_{Det_i}, \dots, K'_{Det_i}$. Then we complete the construction of the composition by mapping $colim_{Det_i}$ into F_i, via φ_i.

Figure 3.2: A diagrammatic representation of how we construct compound compositions: (a) demonstrates how we get the content of a determinate compound composition, whereas (b) demonstrates how we get the content of a determinable compound composition.

commutes. This gives us an important

Principle 3.1. *If for every $\varphi_i, \varphi_j \in \mathrm{Content}(Comp)$ we have that $\varphi_i \sim \varphi_j$ – and therefore that the cardinality of $\mathrm{Content}(Comp)/\sim$ is equal to 1 – then $Comp$ is* **Determinate** *with respect to $\sim$.*

It is clear why $Comp$ should be considered determinate, since a determinate composition can be thought of as a determinable composition whose content contains one structure morphism. Thus, if we have a determinable composition K with $\mathrm{Content}(K) = \Phi$, then for the resulting $\Phi/\sim$, if $card(\Phi/\sim) = 1$ then all the determinations of K are equivalent. Therefore it makes sense to say that K is determinate with respect to $\sim$.

Let us see an example to make this idea clear.

Example 3.4. Words.

Consider the discussion of letters in Example 3.1 above. At the next level of composition, we can construct words as compound compositions over an alphabet $\mathbf{B}$ of letters, where a composition accumulator $\mathcal{A}$ collects letters. The codomain of such a compound composition can be a total order T, since a word is specified by the ordering of its letters. So for each letter $l \hookrightarrow \mathcal{A}$, we would map it onto a single element $t \in T$.[11]

Since letters are determinable, where each determination is a realization of the letter's topology in three-dimensional space, a word is likewise determinable. Thus we can define a word as a compound composition

$$aWord : \mathbf{Determinable}\big(\{\mathcal{A}_i \xrightarrow{\varphi_i} T\}_{i \in I}\big),$$

where each φ_i corresponds to a specific spatial realization of the letters in $\mathcal{A}$.

However, since a word is specified only with regard to how its letters are ordered, then $aWord$ should in a sense be determinate, since each determination realizes the same ordering of the letters of $\mathcal{A}$. This is what the equivalence relation $\sim$ on $\mathrm{Content}(aWord)$ determines. Since for each letter $l \hookrightarrow \mathcal{A}$, there are epimonic morphisms $f : l_{Det_i} \to l_{Det_j}$ between all determinations of l, this means also that any determinations $\varphi_p : \mathcal{A}_p \to T$, $\varphi_q : \mathcal{A}_q \to T$ are equivalent with respect to $\sim$. Therefore $aWord$ is **Determinate** with respect to $\sim$.

3.3.3 Composition Constructors

SUMMARY. We provide a formal definition of a *composition constructor*. First, however, we formalize Cases 1 through 4 from the previous Section 3.3.2.2. In each of these cases, the material $\mathcal{A}$ of each compound composition was fixed, even though the determinations of $\mathcal{A}$ were able to vary. We should also consider the case where the lower level compositions of the compound composition can vary. This is because we may have some compositions that can

[11] Note that this map is epic, since everything in l maps down to the single point t.

gain and lose material, while still remaining 'themselves' in some sense. For example, an oxygen atom can gain and lose neutrons, while still remaining an oxygen atom. Therefore we will also define compound compositions whose lower level compositions vary.

— § —

Recall that $\mathfrak{B}$ is a basis of compositions translated into structures, and for a diagram $\mathcal{D} : J \to \mathfrak{B}$, we call $\mathcal{A} = colim(\mathcal{D})$ a *composition accumulator* over $\mathfrak{B}$. For sake of distinguishing between levels of composition, call $\mathcal{A}$ the *material* of the compound composition (of which $\mathcal{A}$ is the domain structure). Also, for the diagram $\mathcal{D} : J \to \mathfrak{B}$, let $J_{|\mathbf{Dt}}$ denote the subset of J consisting of all j such that $\mathcal{D}(j)$ is a determined composition, and let $J_{|\mathbf{Db}}$ denote the subset of J consisting of all j such that $\mathcal{D}(j)$ is a determinable composition. Since the content of a composition is the most important part of its definition, we will first define a *content constructor*. First, we will define a content constructor whose domain (material) $\mathcal{A}$ remains fixed. This will then allow us to define a content constructor whose domain (material) is variable.

First, we define a content constructor with fixed material $\mathcal{A}$.

Definition 3.2 ($\mathcal{A}$-Content Constructor)**.** Let $\mathcal{D} : J \to \mathfrak{B}$ be a diagram and $\mathcal{A} = colim(\mathcal{D})$. An $\mathcal{A}$-*content constructor* C (over $\mathfrak{B}$) is a pair $C = (D, \Phi)$ where:

1. D is one of the symbols **Determinate** or **Determinable**.
2. Φ depends on $\mathcal{A}$ and D as follows:
   (a) If $\mathcal{A}$ is **Determinate** (meaning that every $\mathcal{D}(j) \hookrightarrow \mathcal{A}$ is **Determinate**), then:
      i. If $D =$ **Determinate**, then $\Phi : \mathcal{A} \to F$ is a structure morphism.
      ii. If $D =$ **Determinable**, then $\Phi = \{\phi_i : \mathcal{A} \to F_i\}_{i \in I}$ is a family of structure morphisms.
   (b) If $\mathcal{A}$ is **Determinable** (meaning that there are $\mathcal{D}(j) \hookrightarrow \mathcal{A}$ that are **Determinable**), then:
      i. If $D =$ **Determinate**, then Φ is a pair $\Phi = (\delta, \psi)$, such that:
         A. For each $j \in J_{|\mathbf{Db}}$, let I_j be the index set of the morphisms $\{\varphi_i\}_{i \in I_j}$ that constitute the content of $\mathcal{D}(j)$. Then for a $j \in J_{|\mathbf{Db}}$ and an $i \in I_j$, we have a structure

$$\mathfrak{Str}\big(N_{j_i}, \mathrm{Content}(\mathcal{D}(j))_i\big).$$

So we need to choose an $i \in I_j$ for each j, and this will determine the selection of determinations from each determinable composition in $\mathcal{A}$. Hence δ will be a map

$$\delta : J_{|\mathbf{Db}} \to \coprod_{j \in J_{|\mathbf{Db}}} \left(\left\{ \mathfrak{Str}\big(N_{j_i}, \mathrm{Content}\big(\mathcal{D}(j)\big)_i\big) : i \in I_j \right\} \right)$$

that sends each $j \in J/\boldsymbol{Db}$ to a determination of $\mathcal{D}(j)$ taken as a structure. Thus δ is a diagram, as is the restriction $\mathcal{D}_{J_{|\mathbf{Dt}}}$ of $\mathcal{D}$ to $J_{|\mathbf{Dt}}$. We thus have their disjoint union $\Delta = \mathcal{D}_{J_{|\mathbf{Dt}}} \amalg \delta$. This then gives a determination of $\mathcal{A}$ as a colimit structure

$$\mathcal{A}_{Det} \xrightarrow[\text{Id}]{} \mathbf{Colimit}(\Delta).$$

B. $\psi : \mathcal{A}_{Det} \to F$ is a structure morphism.

ii. $D = \mathbf{Determinable}$, then Φ is a pair $\Phi = (U, \psi)$, where:

A. For each $j \in J_{|\mathbf{Db}}$, let I_j be the index set of the morphisms $\{\varphi_i\}_{i \in I_j}$ that constitute the content of $\mathcal{D}(j)$. Then U is a subset

$$U \subseteq \prod_{j \in J_{|\mathbf{Db}}} \left(\left\{ \mathfrak{Str}\big(N_{j_i}, \text{Content}\big(\mathcal{D}(j)\big)_i\big) : i \in I_j \right\} \right)$$

of the set of all tuples u whose components u_j consist of a determination of $\mathcal{D}(j)$ taken as a structure. Therefore a tuple $u \in U$ specifies a determination of $\mathcal{A} = colim(\mathcal{D})$, since each determinable composition $\mathcal{D}(j) = K \hookrightarrow \mathcal{A}$ is given a determination by the u_j component of u. Specifically, we get a diagram δ_u, like the δ above, for each $u \in U$; i.e., a diagram

$$\delta_u : J_{|\mathbf{Db}} \to \coprod_{j \in J_{|\mathbf{Db}}} \left(\left\{ \mathfrak{Str}\big(N_{j_i}, \text{Content}\big(\mathcal{D}(j)\big)_i\big) : i \in I_j \right\} \right)$$

that thus determines a structure

$$\mathcal{A}_{Det_u} \xrightarrow[\text{Id}]{} \mathbf{Colimit}(\mathcal{D}_{J_{|\mathbf{Dt}}} \amalg \delta_u).$$

B. $\psi : U \to \mathbf{Str}_1$ is a mapping sending each $u \in U$ to a structure morphism $\eta : \mathcal{A}_{Det_u} \to F$.

Thus Φ determines the structure morphism(s) of the content of a compound composition.

An $\mathcal{A}$-concept constructor is also called a *fixed material content constructor*, since each determination uses some determination of $\mathcal{A}$ for its domain. The reader can confirm that points 2.(a).i, 2.(a).ii, 2.(b).i, and 2.(b).ii correspond, respectively, to Cases 1, 2, 3, and 4 from Section 3.3.2.2.

In contrast to a fixed material content constructor, a *variable material content constructor* is a content constructor where the material $\mathcal{A}$ varies. Thus we provide the following definition.

Definition 3.3 (Variable Material Content Constructor)**.** A *variable material content constructor C (over $\mathfrak{B}$)* is a pair $(\mathbb{D}, \alpha)$, where:

1. $\mathbb{D} = \big\{\mathcal{D}_i : J_i \to \mathfrak{B}\big\}_{i \in I}$ is a collection of diagrams, such that for any distinct $i, k \in I$, we require $J_i \neq J_k$.

2. $\alpha : \mathbb{D} \to \mathfrak{C}$ is a map assigning to each diagram $\mathcal{D} \in \mathbb{D}$, a *colim*$(\mathcal{D})$-content constructor C, where $\mathfrak{C}$ is the collection of all fixed material content constructors over $\mathfrak{B}$.

To get the content of a variable material content constructor, just take the union of all contents from the fixed material content constructors in $\alpha(\mathbb{D})$. For instance, for an $\mathcal{A}$-content constructor C, let $\phi(C)$ denote its content (that is, its set of determinations). If the K-component of C is determinate, then there is only one determination d, so we can wrap it with a singleton $\big\{d\big\}$. Thus we get the content of a variable material content constructor $V = (\mathbb{D}, \alpha)$ as

$$\mathrm{Con}(V) = \bigcup_{\mathcal{D} \in \mathbb{D}} \phi(\alpha(\mathcal{D})). \tag{3.38}$$

Now we can define a *composition constructor* over a basis $\mathfrak{B}$ as follows:

Definition 3.4 (Composition Constructor)**.** A *composition constructor K* over a basis $\mathfrak{B}$ is a triple $K = (N, V, C)$, where:

1. N is a finite string of symbols from $\mathfrak{N}$, used to construct the *name* of K.

2. V is either the symbol **Fixed** or **Variable**. V is called the *variability type* of K.

3. C depends on V as follows:

   (a) If $V =$ **Fixed**, then C is a *fixed material content constructor.*

   (b) If $V =$ **Variable**, then C is a *variable material content constructor.*

   C is called the *content constructor* of K.

This data thus provides what is necessary to create a compound composition over $\mathfrak{B}$. To complete the construction, we see whether the content constructor C generates one structure morphism or more than one; if the former, then K is **Determinate**; if the latter, then K is **Determinable**. Let $\mathrm{Con}(C)$ be the resulting structure morphism or collection of structure morphisms generated by C. Then we get K as a composition by

$$N : \mathbf{D}\big(\mathrm{Con}(C)\big),$$

where $\mathbf{D}$ is either **Determinate** or **Determinable** depending on the value of $\mathrm{Con}(C)$. Note that if $V =$ **Variable**, then K is automatically **Determinable**.

In format, a compound composition is like an elementary composition. The difference is that, for a compound composition, the domain of the content consists of material that is to be formed in some way, whereas there is not yet a notion of material in an elementary composition. Nonetheless, in both situations, the content consists of a structure morphism or a family of structure morphisms.

3.3.4 Super Domain of Compositions

Summary. We briefly define the concept of a *super domain of compositions.*

— § —

A super domain of compositions is just a collection of compositions over some basis. To be concise, we abbreviate 'super domain of compositions' to 'super domain'. We thus define a super domain formally as follows:

Definition 3.5 (Super Domain)**.** A *super domain (over $\mathfrak{B}$)* is a set Σ of compound compositions over $\mathfrak{B}$.

When discussing a super domain Σ, when we refer to the basis, it is convenient to think of this reference as being either to the set of *compositions*, or the set of compositions encoded as *structures.* Likewise, when we refer to the super domain of compositions, we may think of this reference as being either to the set of composition *constructors*, the set of *compositions*, or the set of compositions encoded as *structures.*

3.3.5 Composition Hierarchies

Summary. This section is the apotheosis of the rest of the groundwork provided in this chapter. It provides us with the simple scheme for defining hierarchical universes of objects.

— § —

For a super domain Σ, we get its basis $\mathfrak{B}$ via

$$Basis(\Sigma) = \mathfrak{B}. \tag{3.39}$$

Definition 3.6 (Composition Hierarchy)**.** A *composition hierarchy* $\mathcal{H}$ is a sequence $\mathcal{H} = (\mathfrak{B}, \Sigma_1, \ldots, \Sigma_n)$, where:

1. $\mathfrak{B}$ is an *absolute* basis of compositions.
2. Each Σ_i is such that $Basis(\Sigma_i) = \Sigma_{i-1}$. This means that the set of compositions constructed from a lower level $i-1$ of compositions is used as the basis for the higher level i of compositions.

A simplified representation of a composition hierarchy is depicted in Figure 3.3.

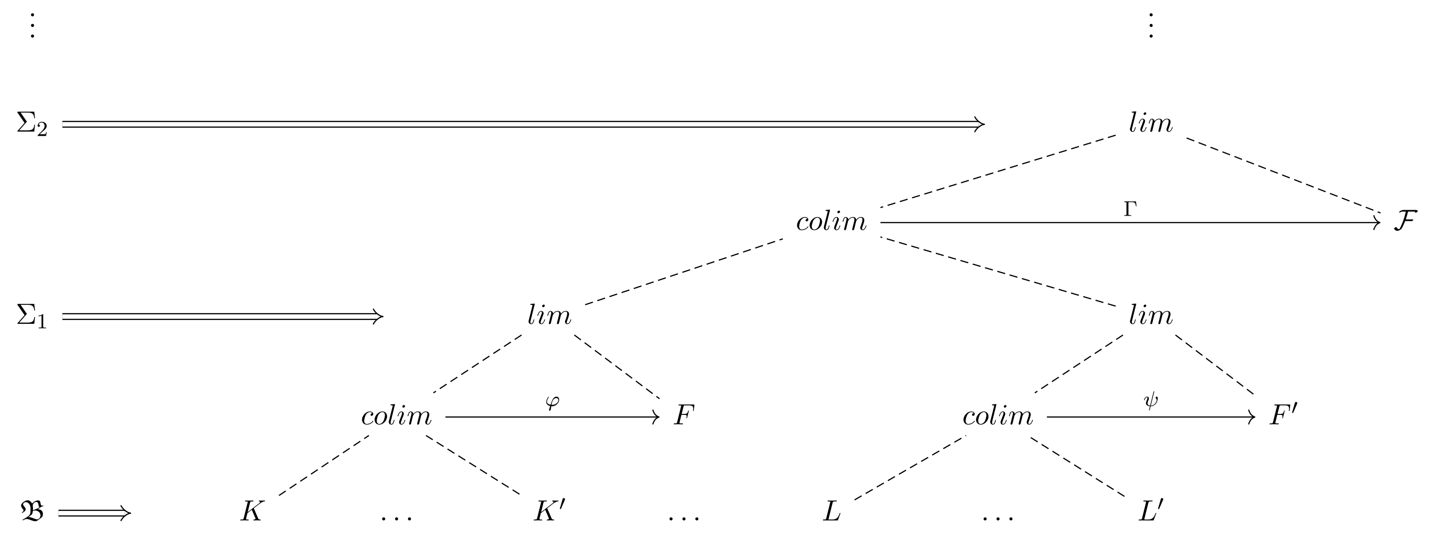


Figure 3.3: A simplified representation of a composition hierarchy. At the bottom of the figure, we have an absolute basis $\mathfrak{B}$ of compositions $K, \dots, K', \dots, L, \dots, L'$. We then come up with combinations of these compositions, and form them via φ and ψ. We then translate the results into limit structures, which forms the super domain Σ_1. And so on.

Chapter 4

Aggregates

4.1 Aggregates

SUMMARY. We provide orientation for the topic of this chapter. We then formally define the concept of an *aggregate*.

— § —

4.1.1 Orientation

SUMMARY. Orientation for the main topic of this chapter is provided.

— § —

In Chapter 3, we defined a compound composition K as a collection of lower level compositions that are given form. This was achieved, roughly, by defining a composition accumulator $\mathcal{A}$, and then mapping $\mathcal{A}$ into some other structure F.[1] We can think of $\mathcal{A}$ as constituting the *material* of K, whereas (the mapping into) F constitutes its *form*.

In this chapter, we will be providing methods for analyzing compositions solely in terms of their material. Such an analysis is useful since it allows for a comparison of compositions based solely on their material constitution, independent of the forms that might differentiate them. For instance, two words related by an anagram are different in form, but in terms of their material (viz. their letters) they are the same. Thus we can compare compositions solely in terms of their parts, and not how those parts are formed into a whole. Such comparisons are made rigorous by the formalisms of this chapter.

What were called *composition accumulators* in Chapter 3 correspond to what we will call *aggregates* in this chapter. Essentially, these are the same kind of object: they both correspond to multisets of lower level compositions. The only difference is formal: whereas

[1]See Section 3.3.2 for a more thorough description.

composition accumulators were defined as diagrams,[2] we will define aggregates in a totally different way.

4.1.2 Definition

Summary. We outline the scheme that we will use to denote aggregates.

— § —

Like the composition accumulators that were presented in Chapter 3, aggregates correspond to multisets of compositions from some basis $\mathbf{B}$. A common formalism for specifying multisets over a universal set X is to define a map $m : X \to \mathbb{N}$, where $m(x) = n > 0$ if x is in the multiset and $m(x) = 0$ if it is not. However, our formalism will make use of a different convention that allows for a more thorough investigation of the relations between multisets. We use the category $\mathcal{P}X$ consisting of subsets of X and inclusions as morphisms. We also equip $\mathcal{P}X$ with the set $\mathbb{N}^+$ of positive natural numbers, and add morphisms $m : U \to \mathbb{N}^+$ for *every* set map from $U \subseteq X$ to $\mathbb{N}^+$. We can then define the following:

Definition 4.1 (Aggregate)**.** Let $\mathbf{B}$ be a basis of compositions. An *aggregate* A is a pair $A = (\mathfrak{sup}, \mathfrak{mul})$, where:

1. $\mathfrak{sup} : 1 \to \mathcal{P}\mathbf{B}$ is a functor from the terminal category. Thus $\mathfrak{sup}$ picks out a subset of $\mathbf{B}$; it is called the *support* of A, and denoted by $\mathfrak{sup}(A)$.

2. $\mathfrak{mul} : \mathfrak{sup}(A) \to \mathbb{N}^+$ is a function that assigns a positive natural number to each element of the support of A; it is called the *multiplicity* of A, and denoted by $\mathfrak{mul}(A)$.

For simplicity, when we refer to the support $\mathfrak{sup}(A)$ of A, we can assume that we are referring to the set $\mathfrak{sup}(A)(1) \in \mathcal{P}\mathbf{B}$.

The reason that we choose this formalism rather than the usual multiset formalism is that this one preserves the distinction between the *support* of a multiset and the *multiplicities.* Thus, rather than checking where a function $m(\mathbf{B})$ is 0 in order to derive the support, we derive it immediately via the functor $\mathfrak{sup}$. Thus for a collection Σ of aggregates, we can investigate them *solely* in terms of *either* $\mathfrak{sup}(\Sigma)$, *or* $\mathfrak{mul}(\Sigma)$, *or* Σ as whole.

Before moving on, we make a comment on terminology. In Section 3.3.4, we defined a *super domain of compositions* as a collection of compositions over a basis. In this chapter, we call a collection of aggregates over a basis a *super domain of aggregates.* For the rest of this chapter, when we speak of super domains, assume that we are talking about super domains of *aggregates.*

[2]I.e., functors $\mathcal{D} : J \to \mathfrak{B}$.

4.2 Categorical Structure of a Super Domain of Aggregates

SUMMARY. The categorical structure of a super domain of aggregates is defined. We also present the notion of a *fiber* of a functor. We then define three functors and discuss their fibers.

— § —

4.2.1 The Category $\mathscr{C}(\Sigma)$ of a Super Domain Σ

SUMMARY. We present the categorical structure of a super domain.

— § —

Given a super domain Σ, we construct a category where the objects are the aggregates $A \in \Sigma$ and where there is a single morphism $\alpha : A \to A'$ if A is a submultiset of A'. Specifically, $\mathscr{C}(\Sigma)$ is the category where:

1. The objects are aggregates $A = (\mathfrak{sup}, \mathfrak{mul})$.
2. The morphisms are pairs $\alpha = (\sigma, \mu) : A \to A'$, where:
   (a) $\sigma : \mathfrak{sup}(A) \hookrightarrow \mathfrak{sup}(A')$ is an inclusion of sets.
   (b) For an aggregate A, consider the function $\mathfrak{mul}(A)$ as the set of pairs (x, n) where $x \in \mathfrak{sup}(A)$ and $\mathfrak{mul}(A)(x) = n$.[3] Then $\mu : \mathfrak{mul}(A) \to \mathfrak{mul}(A')$ sends the pair (x, n) to (x, m) iff $n \leq m$. If there exists a pair $(y, k) \in \mathfrak{mul}(A)$ and $(y, l) \in \mathfrak{mul}(A')$ such that $l < k$, then such a function μ *does not exist*, and thus there is no morphism $\alpha : A \to A'$ in $\mathscr{C}(\Sigma)$.

The category $\mathscr{C}(\Sigma)$ of an arbitrary super domain Σ will be used frequently throughout this chapter. In particular, we will investigate many properties of Σ by evaluating functors from $\mathscr{C}(\Sigma)$ into other categories. These functors are defined in the following section.

4.2.2 A Few Functors and Their Fibers

SUMMARY. We define the notion of a fiber in category theory, and then present a few functors that will be used throughout this chapter.

— § —

[3] Note that this is just the way to define a function as a subset of a Cartesian product; in particular, $\mathfrak{mul} \subseteq \mathfrak{sup}(A) \times \mathbb{N}^+$.

The categories and functors presented in this section will be used throughout this chapter in order to gauge the information about a super domain. Before discussing these categories and functors, we provide a definition of a fiber of a functor, analogously to the definition of a fiber of a function.[4]

Definition 4.2 (Fiber of a Functor)**.** Let $F : C \to D$ be a functor. We define the *fiber* of an object d in D and a morphism g in D as follows:

- For $d \in D_0$, its fiber $F^{-1}(d)$ is the set
$$F^{-1}(d) = \{c \in C_0 : F(c) = d\}.$$
- For $g \in D_1$, its fiber $F^{-1}(g)$ is the set
$$F^{-1}(g) = \{f \in C_1 : F(f) = g\}.$$

In a moment we will present some functors and their fibers. But first, we present the category **FinOrd** of finite ordinals. This category has natural numbers for objects, and a single morphism $+k : n \to m$ whenever $n \leq m$ and $n + k = m$.

(I) *The Rank Functor.* For an arbitrary set X, we define the functor
$$\mathcal{R} : \mathcal{P}X \to \mathbf{FinOrd} : U \mapsto card(U). \tag{4.1}$$
This functor provides a categorical version of the 'rank function' defined on a graded partially ordered set. A graded partial order P is a partial order equipped with a rank function $\rho : P \to \mathbb{N}$ satisfying the following properties:

1. *If $x \prec y$ then $\rho(x) < \rho(y)$.*
2. *If y covers x then $\rho(y) = \rho(x) + 1$.*

The covering relation $\prec\!\!\cdot$ between x and y means that $x \prec y$ and there is no z for which $x \prec z \prec y$.

The idea of a graded partial order P is that there are multiple 'tiers' of elements of P, and each of these tiers can be identified with a positive integer. A power set $\mathcal{P}X$ ordered by inclusion is a graded partial order, and its rank function sends each $U \subseteq X$ to its cardinality. The functor $\mathcal{R}$ is precisely such a rank function; the only difference is that it *also* maps inclusions $i : U \hookrightarrow V$ in $\mathcal{P}X$ to the function $+n : card(U) \to card(V)$, where $n = card(V) - card(U)$.

The fiber of each ordinal $k \in \mathcal{R}(\mathcal{P}X_0)$ is therefore the subset
$$X_k = \{U \in \mathcal{P}X : card(U) = k\}.$$
This of course endows $\mathcal{P}X$ with an equivalence relation $\sim$ where $U \sim V$ iff $card(U) = card(V)$.

[4]For a function $f : X \to Y$, the fiber of f over an element $y \in Y$ is the set $f^{-1}(y) = \{x \in X : f(x) = y\}$.

$$\begin{tikzcd}
 & \underline{\mathcal{P}X} & & & \underline{\mathbf{FinOrd}} \\
 & X \arrow[rrr, Rightarrow] & & & 3 \\
 & & {} \arrow[r, Rightarrow] & {} & \\
\{a,b\} \arrow[uur, hook] & \{a,c\} \arrow[uu, hook] & \{b,c\} \arrow[uul, hook] \arrow[rr, Rightarrow] & & 2 \arrow[uu, "+1"'] \\
 & & & {} \arrow[r, Rightarrow] & {} \\
\{a\} \arrow[uu, hook] \arrow[uur, hook] & \{b\} \arrow[uul, hook] \arrow[uur, hook] & \{c\} \arrow[uul, hook] \arrow[uu, hook] \arrow[rr, Rightarrow] & & 1 \arrow[uu, "+1"'] \\
 & & {} \arrow[r, Rightarrow] & {} & \\
 & \varnothing \arrow[uul, hook] \arrow[uu, hook] \arrow[uur, hook] \arrow[rrr, Rightarrow] & & & 0 \arrow[uu, "+1"']
\end{tikzcd}$$

Figure 4.1: Visualization of the functor from a power set $\mathcal{P}X$ to **FinOrd**.

Remark 4.1. If X is infinite, then there can be no rank function defined from $\mathcal{P}X$ to $\mathbb{N}$. This is because there is no $n \in \mathbb{N}$ that can provide the cardinality of X itself. However, we can define a partial function $\pi : \mathcal{P}X \to \mathbb{N}$ that operates on $U \in \mathcal{P}X$ iff U is finite. We can then say that $\mathcal{P}X$ is *partially graded*, and is therefore a *partially graded partial order.* However, for the remainder of the chapter, assume that when we refer to an arbitrary super domain Σ over an arbitrary basis $\mathbf{B}$, that $\mathbf{B}$ is finite.

Example 4.1. Let $X = \{a, b, c\}$. Then the functor $\mathcal{R} : \mathcal{P}X \to \mathbf{FinOrd}$ can be visualized as in Figure 4.1. According to the figure, we have the following fibers (equivalence classes):

- $\mathcal{R}^{-1}(0) = \{\varnothing\}$
- $\mathcal{R}^{-1}(1) = \{\{a\}, \{b\}, \{c\}\}$
- $\mathcal{R}^{-1}(2) = \{\{a, b\}, \{a, c\}, \{b, c\}\}$
- $\mathcal{R}^{-1}(3) = \{X\}$

(II) *The Support Functor.* This ‘forgetful-like’ functor

$$\mathcal{S} : \mathscr{C}(\Sigma) \to \mathcal{P}\mathbf{B} \tag{4.2}$$

maps aggregates A in $\mathscr{C}(\Sigma)$ to their support sets. $\mathcal{S}$ maps morphisms $\alpha : A \to A'$ in $\mathscr{C}(\Sigma)$ to the inclusion $\mathcal{S}(\alpha) : \mathfrak{sup}(A) \to \mathfrak{sup}(A')$ of their supports.

The fiber of a set $S \in \mathcal{P}\mathbf{B}$ is the set of aggregates $\mathcal{A} \in \mathscr{C}(\Sigma)$ that have the same support.

From $\mathcal{S}$, we derive the next functor.

(III) *The Rank-Support Functor.* This functor is defined as

$$\mathcal{R}_* = \mathcal{R} \circ \mathcal{S} : \mathscr{C}(\Sigma) \to \mathbf{FinOrd}. \tag{4.3}$$

It assigns to each aggregate $A \in \Sigma$ the rank of its support, as demonstrated by the following commutative diagram:

$$\begin{tikzcd}
\mathcal{P}\mathbf{B} \arrow[rr, "\mathcal{R}"] & & \mathbf{FinOrd} \\
& & \\
\mathscr{C}(\Sigma) \arrow[uu, "\mathcal{S}"] \arrow[uurr, dashed, "\mathcal{R}_*=\mathcal{R}\circ\mathcal{S}"'] & &
\end{tikzcd}$$

The fiber of an element n in **FinOrd** is the set of aggregates whose supports have the same cardinality.

Note 4.1. The functors $\mathcal{S}$ and $\mathcal{R}_*$ determine partitions on Σ, as determined by the fibers of their codomains discussed above. These partitions will feature frequently throughout this chapter. We denote the partition defined by $\mathcal{S}$ as $\Sigma/\mathcal{S}$ and that defined by $\mathcal{R}_*$ as $\Sigma/\mathcal{R}_*$. Note that $\Sigma/\mathcal{S}$ is a refinement of $\Sigma/\mathcal{R}_*$.

4.3 Analysis of Super Domains

Summary. A thorough analysis of super domains is performed. We first analyze a super domain in terms of the supports of its aggregates, and then the multiplicities of its supports. This section is meant to provide the reader with an analytical toolkit for investigating aggregates.

— § —

4.3.1 Analysis via $\mathfrak{sup}$

Summary. We present analytical tools for investigating aggregates in terms of their supports.

— § —

Given a super domain $\mathscr{C}(\Sigma)$, our current task is to evaluate its structure[5] and properties with respect to the support sets of its aggregates. Much of this evaluation can be achieved with the functors $\mathcal{S}$ and $\mathcal{R}_*$ discussed in Section 4.2.2. Specifically, we will be investigating $\mathscr{C}(\Sigma)$ for the following reasons:

1. To gauge aggregates for the distribution of the ranks of their supports. This is the question of how many aggregates from Σ have support of rank n, for n a natural number.

2. To investigate the prevalence of a basis composition in Σ, which is to say how frequently a basis composition occurs in Σ. This question will be broken up into two sub-questions:

   (a) How prevalent a basis composition is *at a particular rank.*

   (b) How prevalent a basis composition is *regardless of rank.*

   The motivation for such a distinction will be provided later.

3. To derive a fleshed out identification of the use of supports from different ranks. This reason contrasts with reason (1) above since this current question deals with the *specific* supports of each rank and how frequently they are used, rather than the purely abstract investigation of the ranks themselves. This question will be broken up into three sub-questions:

   (a) How many of the supports from a given rank occur in Σ.

   (b) How many times a *particular* support occurs in Σ.

   (c) The distribution of the use of *all* the supports from a given rank; that is, how many times does each support $s_1, \ldots, s_k$ from rank n occur in Σ.

4.3.1.1 Rank Distribution

Summary. We define the rank distribution of a super domain. The rank distribution specifies how many aggregates with supports of cardinality k occur in the super domain.

— § —

Given a super domain Σ, we want to know how many $A \in \Sigma$ have supports of cardinality n for n between 0 and the cardinality of the basis $\mathbf{B}$. The cardinality of the support of an aggregate A is given by the functor $\mathcal{R}_*(S)$ (see Equation 4.3). Hence for the functor $\mathcal{R}_* : \mathscr{C}(\Sigma) \to \mathbf{FinOrd}$, the fiber of an ordinal n in $\mathbf{FinOrd}$ is the subset $T \subseteq \Sigma$ consisting of all $A \in \Sigma$ such that $card(\mathfrak{sup}(A)) = n$. We take the cardinality of T to derive the

[5]We mean structure in the general sense of the word.

quantity of aggregates in Σ that are composed from the same quantity of unique basis compositions.

For instance, if we have for a super domain of compositions a collection of words W over some alphabet $\mathfrak{A}$ as basis, we may wish to class together all words $U_k \subset W$ such that each $u \in U_k$ has k letters, and is therefore composed from a subalphabet $\mathfrak{A}_i^k \subset \mathfrak{A}$ of cardinality k. We can say that U_k is the subset of W constituted by the set K of k-element subalphabets of $\mathfrak{A}$. This set U_k would correspond to a collection $\mathcal{A}_k$ of aggregates over the basis $\mathfrak{A}$, given by the fiber of $\mathcal{R}_*$ over k in **FinOrd**. The cardinality of the fiber tells us how many words in W have precisely k *unique* letters (thus it ignores duplications of letters). We then let k vary within the range 0 to $card(\mathfrak{A})$, taking the cardinality of the fiber of each such k in order to derive the quantity of words in W that are constituted by k unique letters. The cardinalities of the fibers from 0 to $card(\mathfrak{A})$ constitutes what is called the *rank distribution* of the super domain W.

More generally, for a super domain Σ over basis $\mathbf{B}$ with $card(\mathbf{B}) = n$, the *rank distribution* of Σ is derived by taking the cardinality of the fiber of k in **FinOrd** for $k \in \{0, \dots, n\}$ under the functor $\mathcal{R}_* : \mathscr{C}(\Sigma) \to \mathbf{FinOrd}$. Supposing Σ and $\mathbf{B}$ are finite, this gives rise to an $(n+1)$-tuple

$$\lambda = \big(card(\mathcal{R}_*^{-1}(0)), \dots, card(\mathcal{R}_*^{-1}(n))\big) \tag{4.4}$$

whose value at λ_i (for i in the range $\{1, \dots, n+1\}$) is the cardinality of the fiber of $i-1$ under $\mathcal{R}_*$.[6] Thus, reading λ from left to right gives the quantity of aggregates $A \in \Sigma$ whose supports have cardinality $0, 1, \dots, n$. (Note that there is at most 1 aggregate whose support has a cardinality of 0, namely the empty aggregate.)

Due to the existence of such rank distribution tuples λ, if we are given a collection $\mathbb{S}$ of (finite) super domains over a (finite) basis $\mathbf{B}$ with $card(\mathbf{B}) = n$, then we have the function

$$\Lambda : \mathbb{S} \to \mathbb{N}^{n+1} \tag{4.5}$$

sending each $\Sigma \in \mathbb{S}$ to its rank distribution tuple.

The existence of such a function can be useful for a number of applications, some of which are the following:

Application 4.1. The fiber of $\mathbb{N}^{n+1}$ under Λ can be used to define an equivalence relation on $\mathbb{S}$, where $\Sigma \sim \Sigma'$ whenever they have the same rank distribution.

Application 4.2. Considering the monoidal structure on $\mathbb{N}^{n+1}$ given by the binary operation $+ : \mathbb{N}^{n+1} \times \mathbb{N}^{n+1} \to \mathbb{N}^{n+1}$, one can compare super domains Σ and Σ' by operating on their rank distributions. To be more general, one can extend $\mathbb{N}^{n+1}$ to the group $\mathbb{Z}^{n+1}$, in order to allow for comparisons such as taking the difference between the rank distributions of Σ and Σ'. For example, we can take the difference of the rank distribution of Σ and Σ' via $\Lambda(\Sigma') - \Lambda(\Sigma)$. We can generalize further and define $\mathbb{Z}^{n+1}$ as a $\mathbb{Z}$-module. This allows

[6]The value at λ_i is the cardinality of the fiber of $i-1$ rather than i since we index tuples starting from 1, while the initial object in **FinOrd** is 0.

us to compare super domains based on the multiplicative relationships between their rank distributions; e.g. for scalar multiplication defined in the usual way, i.e.,

$$\cdot : \mathbb{Z} \times \mathbb{Z}^{n+1} \to \mathbb{Z}^{n+1} : (j, v) \mapsto j \cdot v = (j \cdot v_1, \ldots, j \cdot v_{n+1}),$$

we may have that $\Lambda(\Sigma) = j \cdot \Lambda(\Sigma')$. Thus the rank distribution of Σ' is a multiple of the rank distribution of Σ. In this setting, a super domain is converted into a linear combination of basis vectors $v_1, \ldots, v_{n+1}$, where each basis vector denotes a rank.

Application 4.3. We can expand $\mathbb{S}$ by expanding each $\Sigma \in \mathbb{S}$ to its power set $\mathcal{P}\Sigma$, and then unioning all such $\mathcal{P}\Sigma$, giving the set

$$\mathbb{S}^{\mathcal{P}} = \bigcup_{\Sigma \in \mathbb{S}} \{\mathcal{P}(\Sigma) : \Sigma \in \mathbb{S}\}.$$

In other words, we expand $\mathbb{S}$ so that it contains every subsuper domain $\Pi \subset \Sigma$ for every $\Sigma \in \mathbb{S}$. Using the equivalence relation from the first bullet point, we will be able to compare super domains Σ and Σ' in more detail. For instance, suppose $\Sigma \nsim \Sigma'$, but it is the case that $\Lambda(\Sigma)_i \leq \Lambda(\Sigma')_i$ for every index i of their vectors. Then this implies that there exists a subset $\Pi \subset \Sigma'$ such that $\Sigma \sim \Pi$. Under the equivalence relation $\sim$ discussed above, we can define this new (looser) relation $\sim_d$ as a *degenerative* equivalence between Σ and Σ'. Thus we can say the $\Sigma \sim_d \Sigma'$ iff for all i in the range 0 to $n + 1$, we have $\Lambda(\Sigma)_i \leq \Lambda(\Sigma')_i$. Stated more succinctly:

$$\Sigma \sim_d \Sigma' \text{ iff } \forall i \in [0, n+1]\big(\Lambda(\Sigma)_i \leq \Lambda(\Sigma')_i\big).$$

We need not require that $\mathbb{S}$ consist only of super domains sharing the same basis. In a more general setting, given a collection $\mathbb{S}'$ of (finite) super domains over (finite) bases, we can take the maximum

$$\max(\{card(Basis(\Sigma)) : \Sigma \in \mathbb{S}'\}) = m \tag{4.6}$$

in order to derive $\mathbb{N}^{m+1}$ (where $Basis(\Sigma)$ gives the basis of Σ). Thus we have

$$\Lambda : \mathbb{S}' \to \mathbb{N}^{m+1}$$

where for any Σ such that $card(Basis(\Sigma)) = n < m + 1$, the last $(m + 1) - n$ components of $\Lambda(\Sigma)$ are all 0.

4.3.1.2 Prevalence of Individual Basis Compositions

Summary. We provide methods for deriving how prevalent a basis composition is in the aggregates of a super domain.

— § —

We now turn to the question of how many aggregates in a super domain contain *at least one* instance of a particular basis composition b. Let Π be a super domain over $\mathbf{B}$. Then for a basis composition $b \in \mathbf{B}$, our current question is how many $A \in \Pi$ have b in their support. We can also generalize this situation to arbitrary subsets $P \subset \Pi$.

(I) *Absolute Prevalence of a Basis Composition.* Our first task is to check whether or not a higher order composition $A \in \Pi$ contains the basis composition $b \in \mathbf{B}$. We can establish this 'checking mechanism' functorially as follows. Let $b : 1 \to \mathcal{P}\mathbf{B}$ be the functor from the terminal category to the singleton set $\{b\}$. This functor picks out the basis composition b from $\mathbf{B}$ (wrapped in a singleton). Next, define the functor $\pi : 1 \to \mathscr{C}(\Pi)$ in order to pick out an aggregate. For convenience, we can refer to $b(1)$ and $\pi(1)$ as b and π, respectively. Using the functor $\mathcal{S}$ defined in Equation 4.2, we get the functor $\mathcal{S} \circ \pi : 1 \to \mathcal{P}\mathbf{B}$ that maps π to its support. We then check whether the hom-set $\mathrm{Hom}_{\mathcal{P}\mathbf{B}}\big(b, \mathcal{S}(\pi)\big)$ is empty or not. If it's empty, then π does not contain b; if it's not empty, then π does contain b. Moreover if it's not empty, then the cardinality of the hom-set is 1, since there is at most one morphism $i : U \to V$ for $U, V \in \mathcal{P}\mathbf{B}$. Our 'checking mechanism' therefore consists of evaluating whether the cardinality of the hom-set $\mathrm{Hom}_{\mathcal{P}\mathbf{B}}\big(b, \mathcal{S}(\pi)\big)$ is 0 or 1. Thus, to get the quantity of aggregates in Π that contain at least one b, we simply let π vary over all of Π, and sum the cardinalities of hom-sets with domain $\{b\}$. This provides the *prevalence function* with inputs $b \in \mathbf{B}$ and a super domain Π over $\mathbf{B}$, defined as follows:

$$\mathscr{P}(b, \Pi) = \sum_{\pi \in \Pi} card\Big(\mathrm{Hom}_{\mathcal{P}\mathbf{B}}\big(b, \mathcal{S}(\pi)\big)\Big). \tag{4.7}$$

(II) *Relative Prevalence of a Basis Composition.* We obviously do not have to let π vary over *all* of $\mathscr{C}(\Pi)$. Instead, we can let π vary over a subset $P \subset \Pi$. Thus we can evaluate the prevalence of B with respect to P. For instance, we may wish to evaluate the prevalence of b with respect to all the aggregates $P_i \subset \Pi$ of rank i.

The prevalence functions discussed in this section provide information regarding how many aggregates in the super domain contain the basis composition b. What the functions do not provide, however, is the total quantity of instances of b in the super domain in general, since this would require information about the multiplicities of the aggregates. We deal with this problem in Section 4.3.2.2.

4.3.1.3 Prevalence of Supports In a Super Domain

Summary. We provide methods for deriving how frequently a specific support occurs in the aggregates of a super domain.

— § —

We now move on to a more thorough investigation of the distribution of the supports of a super domain Σ.

(I) *How Many k-Element Supports Occur In Σ.* For a super domain Σ over $\mathbf{B}$, we want to know how many of the k-element subsets of $\mathbf{B}$ are used. For $card(\mathbf{B}) = n$, the quantity of k-element subsets of $\mathbf{B}$ is given by

$$\binom{n}{k} = \frac{n!}{k!(n-k)!}.$$

We get the set of aggregates in Σ that have supports with cardinality k by taking the fiber $\mathcal{R}_*^{-1}(k) = \Sigma_k$. We then take the value of the functor $\mathcal{S}(\Sigma_k)$ in order to derive a subset of k-element subsets in $\mathcal{P}\mathbf{B}$, which is given by $\mathcal{R}^{-1}(k)$. We thus take the cardinality $card(\mathcal{S}(\Sigma_k))$ in order to derive the quantity of k-element supports that occur in Σ.

(II) *How Many Times a Particular Support Occurs In Σ.* The next problem is to determine how many times a particular set $B \subseteq \mathbf{B}$ is used as the support for a aggregate $A \in \Sigma$. We can define this as the cardinality of the fiber of B under $\mathcal{S}$. However, we can also solve this problem with similar methods discussed in Section 4.3.1.2. Though this approach is more abstract, it suggests some interesting possibilities, discussed in the following.

First, we pick out a set B by defining it as a functor $B : 1 \to \mathcal{P}\mathbf{B}$, and let $\pi : 1 \to \mathscr{C}(\Sigma)$ vary over the aggregates in Σ. We want to search for all $\pi \in \Sigma$ such that $\mathrm{Hom}_{\mathcal{P}\mathbf{B}}\big(B, \mathcal{S}(\pi)\big) = \{\mathrm{Id}_B\}$. The satisfaction of this condition means that B is the support of π. Thus, to check whether this condition is satisfied, we take the intersection $\mathrm{Hom}_{\mathcal{P}\mathbf{B}}\big(B, \mathcal{S}(\pi)\big) \cap \{\mathrm{Id}_B\}$. Letting $\mathcal{P}\mathbf{B}_1$ denote the set of morphisms in $\mathcal{P}\mathbf{B}$, our 'checking mechanism' is derived as the the following pullback (in **Set**):

$$\begin{tikzcd}
\mathrm{Hom}_{\mathcal{P}\mathbf{B}}\big(B, \mathcal{S}(\pi)\big) \times_{\mathcal{P}\mathbf{B}_1} \{\mathrm{Id}_B\} \arrow[r] \arrow[d] \arrow[dr, phantom, "\lrcorner", very near start] & \{\mathrm{Id}_B\} \arrow[d, hook] \\
\mathrm{Hom}_{\mathcal{P}\mathbf{B}}\big(B, \mathcal{S}(\pi)\big) \arrow[r, hook] & \mathcal{P}\mathbf{B}_1
\end{tikzcd}$$

Thus, in order to derive the total quantity of aggregates $\pi \in \Sigma$ that have support B, we sum the cardinalities of the pullbacks. Setting $\Pi = \Sigma$ (to avoid notational confusion), we get the following function:

$$\Phi(B, \Pi) = \sum_{\pi \in \Pi} card\big(\mathrm{Hom}_{\mathcal{P}\mathbf{B}}\big(B, \mathcal{S}(\pi)\big) \times_{\mathcal{P}\mathbf{B}_1} \{\mathrm{Id}_B\}\big). \tag{4.8}$$

Formalizing the 'checking mechanism' as a pullback is suggestive. For instance, suppose we want to extend the set $\{\mathrm{Id}_B\}$ by allowing it to contain another morphism i, where

$\mathrm{Hom}_{\mathcal{P}\mathbf{B}}\big(B, i(B)\big)$ is non-empty. We might want to do such a thing under a circumstance where the extra compositions in $i(B)$ are in some sense 'extraneous', so an aggregate with support $i(B)$ really just consists of compositions from B. We can define such an extension with a set I of morphisms with domain B as

$$\mathrm{Ext}(B, I) = \{\mathrm{Id}_B\} \cup \bigcup_{i \in I} \mathrm{Hom}_{\mathcal{P}\mathbf{B}}\big(B, i(B)\big). \tag{4.9}$$

We then get the function

$$\Phi'(B, I, \Pi) = \sum_{\pi \in \Pi} card\Big(\mathrm{Hom}_{\mathcal{P}\mathbf{B}}\big(B, \mathcal{S}(\pi)\big) \times_{\mathcal{P}\mathbf{B}_1} \mathrm{Ext}(B, I)\Big). \tag{4.10}$$

Application 4.4. Suppose we have a basis $\mathbf{B}$ consisting of musical pitches along with an $r \in \mathbf{B}$ that denotes silence. Let $P \subset \mathbf{B}$ denote a collection of pitches. We compose a collection M of melodies each of which uses all of the pitches from P. Thus M corresponds to a subset of aggregates $A_P \subset \Sigma$. However, considering that some of these melodies may have rest durations, and thus contain the basis composition r, then this means that there may be $a \in A_P$ such that $\mathcal{S}(a) \neq P$, since it could be the case that $\mathcal{S}(a) = P \cup \{r\}$. If this is the case, then $\Phi(P, \Sigma)$ will not account for some $a \in A_P$, under the condition that $r \in \mathcal{S}(a)$. Thus we would want to extend P by adding the morphism $i^r : P \hookrightarrow P \cup \{r\}$, giving $\mathrm{Ext}(P, \{i^r\})$. Once this is achieved, $\Phi'(P, \{i^r\}, \Sigma)$ will account for all melodies that are composed with all of the pitches from P.

(III) *The Distribution of k-Element Supports In* Σ. For a given rank (cardinality) k of $\mathcal{P}\mathbf{B}$, we wish to determine (a) which sets of rank k are used by Σ, and (b) how many times each set is used as a support of an $A \in \Sigma$. To get the result of (a), we simply take $\Phi(K, \Sigma)$ for $K \subseteq \mathbf{B}$ of rank k. To answer (b), we need to get the result of $\Phi(B, \Sigma)$ for every B of rank k. Then for rank k, we have the quantity of aggregates $A \in \Sigma$ whose support is B for $B \in \mathcal{R}^{-1}(k)$.

To represent this information as a tuple, as we did for the rank distribution in Section 4.3.1.1, we must impose a total order on the set of k-element subsets of $\mathbf{B}$. We can derive this total order by first defining a total order on $\mathbf{B}$ itself. Suppose that $\mathbf{B}$ is totally ordered, and $\mathbf{B}_i$ is the value of the ith position of $\mathbf{B}$. We can construct the totally ordered set of k-element subsets of $\mathbf{B}$ by defining the following algorithm consisting of k nested 'for-loops':

For instance, if we have $\mathbf{B} = \{a, b, c, d\}$ totally ordered as $\langle a, b, c, d\rangle$, and we wish to get the total order of all of its 3-element subsets, we follow the algorithm and get the ordered set

$$K = \langle \{a, b, c\}, \{a, b, d\}, \{a, c, d\}, \{b, c, d\}\rangle. \tag{4.11}$$

Therefore given a total order on $\mathbf{B}$, we have a canonical method of totally ordering the set of k-element subsets of $\mathbf{B}$.

Algorithm 1 Total order of k-element subsets of B

```
$K = \langle\rangle$
**for** $i \in I$ **do**
    **for** $j \in I - \{1\}$ **do**
        $\vdots$
        **for** $l \in I - \{1, \ldots, card(\mathbf{B}) - (card(\mathbf{B}) - k)\}$ **do**
            $K$.append($\{\mathbf{B}_i, \mathbf{B}_j, \ldots, \mathbf{B}_n\}$)
```

Using K above (4.11), if we are given a super domain Σ over $\mathbf{B}$, we can map its totally ordered set K of 3-element subsets to a point $x \in \mathbb{N}^4$ whose ith component consists of the quantity of aggregates $A \in \Sigma$ whose support is the ith position of K. For instance, if $x = (3, 0, 1, 1)$, then Σ has $3, 0, 1,$ and 1 aggregates whose supports are, respectively, $\{a, b, c\}, \{a, b, d\}, \{a, c, d\}$, and $\{b, c, d\}$.

Given a set $\mathbb{S}$ of super domains over $\mathbf{B}$, with $\mathbf{B}$ totally ordered and $card(\mathbf{B}) = n$, we can define

$$\Xi_k : \mathbb{S} \to \mathbb{N}^{\binom{n}{k}}$$

sending each $\Sigma \in \mathbb{S}$ to its tuple representing the quantity of $A \in \Sigma$ that have a specific k-element support. We can refer to such a function as a 'k-distribution'. This function is in some sense opposite to the Λ function from Equation 4.4, which provides the rank distribution. Whereas Λ provides information regarding the 'vertical' distribution of supports of Σ, Ξ_k provides information regarding their 'horizontal' distribution. In general, the 'horizontal' situation is messier than the 'vertical' situation. For instance, we need $card(\mathbf{B}) + 1$ functions in order to define $\Xi_k(\mathbb{S})$ for k in the range 0 to $card(\mathbf{B})$. We also therefore require a unique domain $\mathbb{N}^{\binom{n}{k}}$ for each k. Furthermore, we require a choice of total order on $\mathbf{B}$, whereas $\mathcal{P}\mathbf{B}$ naturally comes equipped with the total order given by the functor $\mathcal{R} : \mathcal{P}\mathbf{B} \to \mathbf{FinOrd}$.

However, the increase in messiness of the horizontal distributions corresponds to an increase in depth of information. Whereas the vertical rank distribution hides all information regarding which particular supports are used by aggregates, the horizontal k-distribution specifies this information precisely. Moreover, the rank distribution can be recovered from the k-distribution by summing all ith components of $\Xi_k(\Sigma)$, letting k vary over all ranks.

An application of the k-distribution is the following:

Application 4.5. Suppose we have $\Sigma, \Sigma' \in \mathbb{S}$ over the (totally ordered) basis $\mathbf{B}$. We can compare their values under Ξ_k in various ways. For instance, if $\Xi_k(\Sigma) \neq \Xi_k(\Sigma')$, yet $\Xi_k(\Sigma')$ can be transformed into $\Xi_k(\Sigma)$ by permuting the values in its vector, then this suggests some kind of relationship between Σ and Σ'. For starters, it means that there is a bijection $f : \mathbf{B} \to \mathbf{B}$ which can be used to change the total order of $\mathbf{B}$ used for Σ into the total order of $\mathbf{B}$ used for Σ'. Denoting this variant of Σ' by Σ'_f, we would then have $\Xi_k(\Sigma) = \Xi_k(\Sigma'_f)$. This defines an equivalence relation on $\mathbb{S}$, where for any $\Sigma, \Sigma' \in \mathbb{S}$, we would have $\Sigma \sim \Sigma'$

if there exists a bijection $f : \mathbf{B} \to \mathbf{B}$ such that $\Xi_k(\Sigma) = \Xi_k(\Sigma'_f)$. We could take this a step further, and denote a stronger equivalence relation whereby $\Sigma \sim \Sigma'$ if for every $i \in \{0, \ldots, card(\mathbf{B})\}$ there exists a $g : \mathbf{B} \to \mathbf{B}$ such that $\Xi_i(\Sigma) = \Xi_i(\Sigma'_g)$. A yet stronger equivalence relation would be if this g was the same for every i. In this situation, there would be a strong symmetry between Σ and Σ', where for every aggregate $A \in \Sigma$, there would be a unique $A' \in \Sigma'$ such that $g(\mathfrak{sup}(A)) = \mathfrak{sup}(A')$. Under such a condition, we could say that Σ and Σ' are weakly[7] automorphic.

4.3.2 Analysis via $\mathfrak{mul}$

SUMMARY. We present analytical tools for investigating aggregates in terms of the multiplicities of their supports.

— § —

We now move on to the analysis of a super domain Σ in terms of the multiplicities of its aggregates. Specifically, we will be investigating the multiplicities of aggregates for the following reasons:

1. To consider under what condition two aggregates can be considered ‘homomorphic’.
2. To evaluate the ubiquity of a basis composition $b \in \mathbf{B}$ in Σ. This contrasts with the task of Section 4.3.1.2, insofar as in that section we only took into account whether or not b was in the support of each $A \in \Sigma$, whereas in the following section we take into account the multiplicity of b in each A.

4.3.2.1 Aggregate Homomorphisms

SUMMARY. We define aggregate homomorphisms. Such homomorphisms are rather unusual compared to the usual homomorphisms that occur elsewhere in mathematics. We demonstrate how aggregate homomorphisms are built up from the more general concept of a ‘local homomorphism’ of aggregates.

— § —

In mathematics, two objects X and Y are called *homomorphic* if, to some extent, they have the same structure. The question of whether or not two objects are homomorphic can only be answered once a uniform context is given for determining the structure of X and Y. For instance, if X and Y are groups, then we can check whether or not X and Y are homomorphic *as groups* with the following criterion:

[7] We say ‘weakly’ since this condition does not yet take into account the multiplicities of Σ and Π.

Definition 4.3 (Group Homomorphism)**.** Two groups G and H, under the binary operations of $+$ and $*$, respectively, are *homomorphic* if there exists a function $\varphi : G \to H$ such that

$$\varphi(a + b) = \varphi(a) * \varphi(b)$$

for every $a, b \in G$.

Now suppose we were to frame G and H in a new light after realizing that the binary operations $\star : G \times G \to G$ and $\diamond : H \times H \to H$ turn G and H into rings. Then φ would still constitute a homomorphism on their underlying groups, yet it might fail to constitute a homomorphism on their ring structure. Thus we would say that G and H are homomorphic *as groups* but not *as rings*. This is why we require a context in order to determine whether or not two objects X and Y are homomorphic. For instance, in the degenerative case where we compare whether or not X and Y are homomorphic *as sets*, the answer is yes in every instance, so long as X and Y are non-empty.

Now, our current task is to establish a homomorphism criterion for aggregates. In the spirit of this chapter, our criterion will only take into account the multiplicities of basis compositions. Also, we will work from a more general homomorphism criterion to a more rigid one. The general criterion is that of 'local homomorphism'; the idea is that two aggregates are 'locally homomorphic' wherever they each have a basis composition sharing the same multiplicity. For instance, if an aggregate A contains three b compositions, while another aggregate A' contains three b' compositions, then A and A' are locally homomorphic at b resp. b'. Let's now move on to the precise formalism.

Let Σ be a super domain over $\mathbf{B}$. Denote by $\mathcal{P}\mathbf{B} + \mathbb{N}^+$ the category consisting of $\mathcal{P}\mathbf{B}$ along with the object $\mathbb{N}^+$, where for any set $X \in \mathcal{P}\mathbf{B}$, the hom-set $\mathrm{Hom}_{\mathcal{P}\mathbf{B}}(X, \mathbb{N}^+)$ consists of all set maps from X to $\mathbb{N}^+$. Recall that an aggregate A is a specified by a pair $(\mathfrak{sup}, \mathfrak{mul})$ where $\mathfrak{sup} : 1 \to \mathcal{P}\mathbf{B} + \mathbb{N}^+$ is a functor picking out a set in $\mathcal{P}\mathbf{B} + \mathbb{N}^+$ (other than $\mathbb{N}^+$ itself) and $\mathfrak{mul} : \mathfrak{sup}(1) \to \mathbb{N}^+$ assigns its multiplicities. For an aggregate A, let $\mathfrak{sup}(A)$ denote its support set and $\mathfrak{mul}(A)$ its multiplicity function. Now we provide the definition of local homomorphism:

Definition 4.4 (Local Homomorphism)**.** Let A and A' be aggregates, and suppose $b \in \mathfrak{sup}(A)$ and $b' \in \mathfrak{sup}(A')$. Then A and A' are *locally homomorphic* at the pair (b, b') iff $\mathfrak{mul}(A)(b) = \mathfrak{mul}(A')(b')$.

Hence for $A, A' \in \Sigma$, we derive all pairs of local homomorphisms (b, b') via the fibered product

$$\begin{tikzcd}
\mathfrak{sup}(A) \times_{\mathbb{N}^+} \mathfrak{sup}(A') \arrow[r, "\mathrm{pr}_2"] \arrow[d, "\mathrm{pr}_1"'] & \mathfrak{sup}(A') \arrow[d, "\mathfrak{mul}(A')"] \\
\mathfrak{sup}(A) \arrow[r, "\mathfrak{mul}(A)"'] & \mathbb{N}^+
\end{tikzcd} \tag{4.12}$$

Example 4.2. Let A and A' be aggregates, where $\mathfrak{sup}(A) = \{a, b, c\}$ and $\mathfrak{sup}(A') = \{p, q\}$, and whose multiplicities are given by the following:

1. $\mathfrak{mul}(A) : \mathfrak{sup}(A) \to \mathbb{N}^+$
   - $a \mapsto 2$
   - $b \mapsto 3$
   - $c \mapsto 3$
2. $\mathfrak{mul}(A') : \mathfrak{sup}(A') \to \mathbb{N}^+$
   - $p \mapsto 4$
   - $q \mapsto 3$

Then we have the set $\mathfrak{sup}(A) \times_{\mathbb{N}^+} \mathfrak{sup}(A') = \{(b, p), (b, q)\}$.

The fibered product construction does not provide us with a homomorphism in the usual sense of the word, since a homomorphism is generally conceived as a function $\varphi : X \to Y$ on the underlying sets of X and Y. Instead, we are given a set of pairings (x, y) that establish *possible* choices for homomorphic correspondences. Thus we define a *partial homomorphism* between aggregates A and A' as follows:

Definition 4.5 (Partial Homomorphism)**.** A *partial homomorphism* between aggregates A and A' is a partial function $\pi : \mathfrak{sup}(A) \to \mathfrak{sup}(A')$ such that the following diagram commutes:

$$\begin{tikzcd}
\mathfrak{sup}(A) \arrow[r, "\pi"] \arrow[dr, "\mathfrak{mul}(A)_{|\mathrm{dom}(\pi)}"'] & \mathfrak{sup}(A') \arrow[d, "\mathfrak{mul}(A')"] \\
& \mathbb{N}^+
\end{tikzcd} \tag{4.13}$$

where $\mathfrak{mul}(A)_{|\mathrm{dom}(\pi)}$ is $\mathfrak{mul}(A)$ restricted to the domain of π.

A partial homomorphism between A and A' therefore corresponds to a subset $\pi \subseteq \mathfrak{sup}(A) \times_{\mathbb{N}^+} \mathfrak{sup}(A')$. This provides an alternative definition:

Definition 4.6 (Partial Homomorphism)**.** A *partial homomorphism* between aggregates A and A' is a subset $\pi \subseteq \mathfrak{sup}(A) \times_{\mathbb{N}^+} \mathfrak{sup}(A')$ such that if $(b, c), (b, c') \in \pi$, then $c = c'$. Therefore π is a partial homomorphism iff it is a *functional relation.*[8]

[8]Recall that a relation $R \subseteq X \times Y$ is called *functional* iff $(x, y), (x, z) \in R$ implies $y = z$, that is, iff an element $x \in X$ appears no more than once in R.

A partial homomorphism π is called *maximal* when $\mathrm{pr}_1(\pi) = \mathrm{pr}_1(\mathfrak{sup}(A) \times_{\mathbb{N}^+} \mathfrak{sup}(A'))$, where pr_1 is projection on the first component. The idea is that π is maximal when as many elements $x \in \mathfrak{sup}(A)$ as possible are mapped by π.

With partial homomorphisms defined, we can now move on to the case of defining a total homomorphism between aggregates. This would correspond to the standard situation in mathematics, where objects X and Y are considered homomorphic if one is a substructure of the other. For instance, a group homomorphism $\varphi : G \to H$ means either that G is isomorphic to a subgroup of H, or that H is isomorphic to a subgroup of G. With the groundwork laid out in the definition of partial homomorphism, we define a total homomorphism as follows:

Definition 4.7 (Total Homomorphism)**.** A *total homomorphism* between aggregates A and A' is a partial homomorphism $\pi : \mathfrak{sup}(A) \to \mathfrak{sup}(A')$ where π is a total function.

The definition of an isomorphism between aggregates naturally follows.

Definition 4.8 (Isomorphism)**.** An *isomorphism* between aggregates A and A' is a total homomorphism $\pi : \mathfrak{sup}(A) \to \mathfrak{sup}(A')$ that is also a bijection.

A unique case of isomorphism is when $\sup(A) = \sup(A') = X$. In this case, we can say that A and A' are *symmetric*, since the function f that makes

$$\begin{tikzcd}
X \arrow[r, "f"] \arrow[dr, "\mathfrak{mul}(A)"'] & X \arrow[d, "\mathfrak{mul}(A')"] \\
& \mathbb{N}^+
\end{tikzcd}$$

commute is a symmetry of X, i.e., a bijection from X to itself.

4.3.2.2 Ubiquity of Individual Basis Compositions

Summary. This section follows the line of thought from Section 4.3.1.2, where we discussed the *prevalence* of individual basis compositions. The difference is that this section takes into account the multiplicities of basis compositions.

— § —

In Section 4.3.1.2, we formalized the prevalence of a basis composition b in the super domain Π. There, the prevalence function merely took into account whether or not b was present in some $A \in \Pi$, but it didn't take into account the multiplicity of b in A. In this section, we take into account the multiplicities of basis compositions in order to derive their ubiquity in Π.

(IA) *Absolute Ubiquity of a Basis Composition.* Let Π be a super domain over $\mathbf{B}$ and suppose $b \in \mathbf{B}$. Define the functor

$$\mu_b : \mathscr{C}(\Pi) \to \mathbf{FinOrd} \tag{4.14}$$

that sends $A \in \Pi$ to the multiplicity of its b component. We can confirm that μ_b is indeed a functor since if $\mathrm{Hom}_{\mathcal{P}\mathbf{B}}(A, A') \neq \varnothing$, then $\mathfrak{mul}(A)(b) \leq \mathfrak{mul}(A')(b)$, and therefore μ_b sends the (unique) morphism $\alpha : A \to A'$ to $+(\mathfrak{mul}(A')(b) - \mathfrak{mul}(A)(b)) : \mathfrak{mul}(A)(b) \to \mathfrak{mul}(A')(b)$. To get the ubiquity of b in Π, we let the functor $\pi : 1 \to \mathscr{C}(\Pi)$ vary over all $A \in \Pi$, summing the values of $\mu_b(A)$. We thus derive the *ubiquity function*

$$\mathscr{U}(b, \Pi) = \sum_{\pi \in \Pi} \mu_b(\pi). \tag{4.15}$$

(IB) *Relative Ubiquity of a Basis Composition.* Like in Section 4.3.1.2, we can let π vary over a subset $P \subset \Pi$. We can thus derive the relative ubiquity of b with respect to P by restricting the Π argument to P. Taking this a step further, we can define a functor

$$\mathscr{U}(b, -) : \mathcal{P}\Pi \to \mathbf{FinOrd} \tag{4.16}$$

that sends each $P \subseteq \Pi$ to $\mathscr{U}(b, P)$. This would provide a method of tracking how ubiquitous b is locally, i.e., with respect to subsets of Π.

We know that $\mathscr{U}(b, -)$ is a functor since it sends inclusions $i : P \hookrightarrow Q$ to the function

$$+(\mathscr{U}(b, Q) - \mathscr{U}(b, P)) : \mathscr{U}(b, P) \to \mathscr{U}(b, Q).$$

Let us see an example of this functor in action.

Example 4.3. Let $\mathbf{B} = \{a, b, c\}$ and $\Pi = \{P, Q, R\}$ defined as follows:

- $P = \{(a, 3), (b, 4), (c, 2)\}$
- $Q = \{(b, 1), (c, 1)\}$
- $R = \{(a, 4)\}$

We then get the correspondence in Figure 4.2 via the functor $\mathscr{U}(b, -)$.

(IIA) *Comparing Absolute Ubiquity of Different Basis Compositions.* We can also compare the ubiquity in Π of different basis compositions $b, \ldots, h \in \mathbf{B}$. We simply let the first argument $\mathscr{U}(-, \Pi)$ vary over the elements of $\mathbf{B}$. We can view $\mathscr{U}(-, \Pi)$ as a functor. First, consider $\mathbf{B}$ as a discrete category with objects $b \in \mathbf{B}$ and only identity morphisms. Then $\mathscr{U}(-, \Pi) : \mathbf{B} \to \mathbf{FinOrd}$ is a functor that assigns to each $b \in \mathbf{B}$ its ubiquity in Π.

$$\begin{tikzcd}
 & \underline{\mathcal{P}\Pi} & & \underline{\mathbf{FinOrd}} \\
 & \Pi \arrow[rr, dash, dashed] & & 5 \\
\{P,Q\} \arrow[ur] \arrow[urrr, dash, dashed] & \{P,R\} \arrow[u] \arrow[rr, dash, dashed, bend left] & \{Q,R\} \arrow[ul] \arrow[ddr, dash, dashed] & 4 \arrow[u] \\
 & & & \vdots \\
\{P\} \arrow[uu] \arrow[uur] \arrow[uurrr, dash, dashed] & \{Q\} \arrow[uul] \arrow[uur] \arrow[rr, dash, dashed, bend left] & \{R\} \arrow[uul] \arrow[uu] \arrow[dr, dash, dashed] & 1 \arrow[uu, bend right] \\
 & \varnothing \arrow[ul] \arrow[u] \arrow[ur] \arrow[rr, dash, dashed] & & 0 \arrow[u]
\end{tikzcd}$$

Figure 4.2: The action of $\mathcal{U}(b,-)$ is shown by the dotted lines. The reader can confirm that $\mathcal{U}(b,-)$ is indeed a functor.

In this situation, we can check whether some b is less (or as) ubiquitous than c if $\mathrm{Hom}_{\mathbf{FinOrd}}\big(\mathcal{U}(b,\Pi),\mathcal{U}(c,\Pi)\big)$ is non-empty, since this means that $\mathcal{U}(b,\Pi) \leq \mathcal{U}(c,\Pi)$.

(IIB) *Comparing Relative Ubiquity of Different Basis Compositions.* Finally, we can synthesize (IB) and (IIA) by defining the functor

$$\mathcal{U}(-,-) : \mathbf{B} \times \mathcal{P}(\Pi) \to \mathbf{FinOrd}. \tag{4.17}$$

The product category $\mathbf{B} \times \mathcal{P}(\Pi)$ has for objects pairs (b, P) for $b \in \mathbf{B}$ and $P \subseteq \Pi$, and morphisms $(\mathrm{Id}_b, i) : (b, P) \to (b, Q)$ where $\mathrm{Id}_b : b \to b$ is a morphism in $\mathbf{B}$ and $i : P \to Q$ a morphism in $\mathcal{P}(\Pi)$. Since $\mathbf{B}$ is discrete, the Id_b-components of the morphisms must be identity morphisms. Therefore $\mathbf{B} \times \mathcal{P}(\Pi)$ essentially creates $card(\mathbf{B})$ disjoint copies of $\mathcal{P}(\Pi)$, each copy being indexed by an element $b \in \mathbf{B}$.

The functor $\mathcal{U}(-,-)$ sends pairs (b, P) to the ubiquity of b with respect to P. In this situation, we can evaluate the ubiquity of b with respect to P in comparison with the ubiquity of c with respect to Q, by checking the hom-set $\mathrm{Hom}_{\mathbf{FinOrd}}\big(\mathcal{U}(b,P),\mathcal{U}(c,Q)\big)$. If it is non-empty, then c with respect to Q is more ubiquitous than b with respect to P. The (necessarily) unique morphism in the hom-set tells us by how much, since it is the function $+n$ for $n = \mathcal{U}(c,Q) - \mathcal{U}(b,P)$. If the hom-set is empty, then we know that b with respect to P is more ubiquitous than c with respect to Q.

We could have also achieved another method of comparison by evaluating the functors $\mathcal{U}(b,\Pi)$ and $\mathcal{U}(c,\Pi)$, and then comparing them via a natural transformation $\eta : \mathcal{U}(b,\Pi) \to \mathcal{U}(c,\Pi)$. This would enable a comparison between b's ubiquity at each $P \in \mathcal{P}(\Pi)$ to c's ubiquity at P.

4.3.3 Examples

SUMMARY. We provide some examples of super domains and analyze their properties.

— § —

Example 4.4. We provide this example in two parts.

(I) *Chemical Elements.* Consider the chemical elements from the periodic table as a super domain *Chem*. In this case, we have as the basis of *Chem* the set *Particles* consisting of protons, neutrons, and electrons. While the number of protons and electrons for a neutral chemical element $c \in Chem$ are known (they are equal to the atomic number), the number of neutrons may vary, although their quantity can in general be calculated by subtracting the atomic number from the atomic mass. For the sake of argument, however, we can ignore the multiplicity assignments on neutrons; thus we will compare elements $c, c' \in Chem$ on the basis of their protons and electrons alone. To do this formally, we can use the following method. Let Σ be a super domain over basis $\mathbf{B}$ with $B \subset \mathbf{B}$. We derive

Σ/B after 'modding out' B, by which we mean removing all data associated with the B basis compositions. We thus get $Particles/\{\text{neutron}\}$ as the super domain that allows us to compare aggregates in $Particles$ based on their proton and electrons distributions alone.

Before moving on, let's define the following property for aggregates:

Definition 4.9 (Homogeneous)**.** An aggregate A is called *homogeneous* when there exists a unique n such that $\mathfrak{mul}(A)(x) = n$ for all $x \in \mathfrak{sup}(A)$.

We know that for a neutrally charged atom A, it contains as many protons as electrons. Therefore every aggregate $C \in Particles/\{neutron\}$ is homogeneous.

(II) *Rank Distribution of Particles.* Of the 118 standard elements of the periodic table, all of them except for hydrogen contain all of the basis compositions of protons, electrons, and neutrons. Hydrogen differs from the others as it does not contain neutrons. Therefore the rank distribution of $Particles$ is given by the tuple $(0, 0, 1, 117)$. $Particles$ contains 0 aggregates with the empty support, 0 aggregates with singleton supports, 1 aggregate with a 2-element support (hydrogen), and 117 aggregates with a 3-element support.

Example 4.5. *Stockhausen's Klavierstücke XI.* This piano piece consists of 19 musical fragments dispersed on a page. The performer is told to start on any one of these fragments, and once finished move to another one at random. The piece is finished after one fragment has been played three times.

Let P denote the set of fragments. A realization of *Klavierstücke XI* corresponds to an aggregate A over P.

For a realization, a $p \in P$ can be played anywhere from 0 to 3 times. If it is played 3 times, then all other $q \in P$ can be played anywhere from 0 to 2 times. A realization of *Klavierstücke XI* therefore corresponds to an aggregate A whose vector representation $v(A) = (a, \ldots, 3, \ldots, k)$ contains one 3 and any number from 0 to 2 for all other components. Let $u \in P$ be the fragment that is played for a third time, thus the fragment that the piece ends on. With respect to aggregates – and not with respect to e.g. the order that the fragments are played – there are $3^{18} = 387,420,489$ realizations that end on u. Since there are 19 fragments in total, and the piece can end on any of them, there are therefore $19 \cdot (3^{18}) = 7,360,989,291$ realizations (in terms of aggregates). Thus the super domain corresponding to realizations of *Klavierstücke XI* has $7,360,989,291$ aggregates.

Example 4.6. We present this example in three parts.

(I) *Databases of Faculty by Department.* In this example, we consider an aggregate over a set $\mathbf{D}$ of university department names. The idea is that a university corresponds to an aggregate over $\mathbf{D}$, given by its quantity of professors in each department $d \in \mathbf{D}$.

Let $String$ be a set of strings of symbols, and let $UniFac$ be the set of faculty members at some university Uni. We can define a set function $dep : UniFac \to String$ that sends each faculty member to his or her corresponding department. For example, for $x \in UniFac$, if we have $dep(x) =$ 'computer science', then x is a computer scientist at Uni. If we encode $String$ and $UniFac$ as structures[9], then this turns $dep : UniFac \to String$ into a structure morphism. With this established, we can define a new structure

$$UniFacDep \underset{\mathrm{Id}}{\longrightarrow} \mathbf{Limit}(UniFac \xrightarrow{dep} String)$$

whose functional 1-addressed points are pairs (x, d) where x is a faculty member at Uni and d his or her corresponding department.

Now, we are currently considering a super domain Σ_{Uni} over $\mathbf{D}$ where $\mathbf{D} \subset String$ is a set of department names *in general*, i.e., regardless of the particular university. For example, we could have $\mathbf{D} = \{$'computer science', 'philosophy', 'music'$\}$. For each university U we would get a unique structure

$$UFacDep \underset{\mathrm{Id}}{\longrightarrow} \mathbf{Limit}(UFac \xrightarrow{dep} String)$$

Now suppose we want to find the set of all faculty members $x \in UFac$ that are in department $d \in \mathbf{D}$. If we let $d : 1 \to String$ denote a functional structure morphism that picks out a department $d \in String$, then we have a diagram

$$\begin{tikzcd} & 1 \arrow[d, "d"] \\ UFac \arrow[r, "dep"'] & String \end{tikzcd}$$

The limit of $\mathcal{D}$ is given by the pullback

$$\begin{tikzcd} UFac \times_{String} 1 \arrow[r, "\mathrm{pr}_2"] \arrow[d, "\mathrm{pr}_1"'] & 1 \arrow[d, "d"] \\ UFac \arrow[r, "dep"'] & String \end{tikzcd}$$

where the functional 1-addressed points of $UFac \times_{String} 1$ are pairs (x, d) where $x \in UFac$ and d is some fixed department. Thus $UFac \times_{String} 1$ corresponds to the structure

$$dAtU \underset{\mathrm{Id}}{\longrightarrow} \mathbf{Limit}(\mathcal{D}).$$

Therefore $dAtU$ consists of all faculty members $x \in UFac$ that are in department d.

[9]See Example 3.2 for how we convert sets into structures.

Given $dAtU$, we want to define an elementary composition whose content specifies a professor in department d at U. For example, if $d =$ 'computer science', we want to define the elementary composition whose content corresponds to 'computer science professor at university U'. We thus define the following composition

$$dProfAtU : \textbf{Determinable}\big(\{1 \xrightarrow{f_i} dAtU\}_{i \in I}\big)$$

where f_i picks out a computer science professor at U. The cardinality of the content of $dProfAtU$ is the quantity of computer science professors at U. Thus, for each $d \in \mathbf{D}$ we would get its multiplicity from the cardinality of the content of such a composition $dProfAtU$. Once we have all the cardinalities of the content of each $dProf@U$ for $d \in \mathbf{D}$, we would get an aggregate over $\mathbf{D}$. For instance, if

$$\mathbf{D} = \{\text{'computer science', 'philosophy', 'music'}\}$$

and U has 8 computer science professors, 5 philosophy professors, and 0 music professors, then we would get an aggregate $A_U = (\{\text{'computer science', 'philosophy'}\}, \mathfrak{mul})$, where $\mathfrak{mul}(\text{'computer science'}) = 8$ and $\mathfrak{mul}(\text{'philosophy'}) = 5$.

For a set $\mathcal{U}$ of universities, we would thus get similar such aggregates over $\mathbf{D}$, thus constituting a super domain $\Sigma_{\mathcal{U}}$. Note that we have a potential problem. If there are multiple universities with the same departmental distributions of faculty, then they constitute the same aggregate over $\mathbf{D}$. For the sake of this example, assume that we allow for a super domain that is a multiset. We can do this by attaching a unique index to each $A \in \Sigma_{\mathcal{U}}$. For instance, if we have two universities $U, V \in \mathcal{U}$ that constitute the same aggregate $A \in \Sigma$, then we can index them with A_U and A_V. This provides a method for distinguishing between two equal aggregates.

Many of the analytical tools discussed in Section 4.3 can be put to use to analyze $\Sigma_{\mathcal{U}}$. We will discuss some of them in what follows.

(II) *Prevalence of University Departments.* In Section 4.3.1.2, our topic was to determine which aggregates in a super domain contain a basis composition b. In the current example, this question corresponds to asking how many universities $U \in \mathcal{U}$ have the department $d \in \mathbf{D}$, assuming that if there are no professors in department d then the department d does not exist. We are provided with the quantity of universities in $\mathcal{U}$ that have a d department via the *prevalence function* $\mathscr{P}(d, \Sigma_{\mathcal{U}})$ from Equation 4.7.

We can also evaluate the relative prevalence of d with respect to a subset $\Pi \subset \Sigma_{\mathcal{U}}$. For instance, suppose that all universities in $\mathcal{U}$ are in the United States, and we want to partition $\mathcal{U}$ according to the states that the universities are in. For US states s and t, let Σ_s and Σ_t denote, respectively, the subsets of $\Sigma_{\mathcal{U}}$ corresponding to all universities that are in the state s and t. We can compare the prevalence of department d with respect to both s and t via the prevalence function. If we have $\mathscr{P}(d, \Sigma_s) = n$ and $\mathscr{P}(d, \Sigma_t) = m$, then we can compare which state, s or t, has more universities with a d department.

(III) *Ubiquity of Professors From a Given Department.* In Section 4.3.2.2, our topic was to determine the total quantity of a basis composition b in a super domain Σ. In the current example, this question corresponds to asking how many d professors there are in total in $\Sigma_{\mathcal{U}}$. This is provided by the *ubiquity function* $\mathcal{U}(d, \Sigma_{\mathcal{U}})$ from Equation 4.15. As we saw in Section 4.3.1.2, the ubiquity function extends to a functor

$$\mathcal{U}(d, -) : \mathcal{P}\Sigma_{\mathcal{U}} \to \mathbf{FinOrd}.$$

In the current context, this functor provides information regarding how many d professors there are with respect to subsets of universities $\mathcal{V} \subseteq \mathcal{U}$.

Application 4.6. A practical use for this functor would be to evaluate how many d professors there are with respect to different geographical locations. For instance, we can associate a subset of universities with the (open) set constituted by the union of their geographical coordinates. Let $G : \mathcal{P}\Sigma_{\mathcal{U}} \to \tau$ be the function mapping subsets of $\Sigma_{\mathcal{U}}$ to the open sets in τ corresponding to their geographical coordinates. When we evaluate $\mathcal{U}(d, -)$ over two arguments $\Sigma_L, \Sigma_K \in \mathcal{P}\Sigma_{\mathcal{U}}$, we are essentially evaluating how many d professors there are in the geographical locations $G(\Sigma_L)$ and $G(\Sigma_K)$. Thus, for two people p, q wishing to study under a professor from department d (say, for a PhD thesis), but p within a geographical location close to some open set $U \in \tau$ and q close to some other $V \in \tau$, we can compare how many different options they would have via $\mathcal{U}(d, G^{-1}(U))$ and $\mathcal{U}(d, G^{-1}(V))$, where G^{-1} maps open sets in τ back to their subsets $\Sigma_L \in \mathcal{P}\Sigma_{\mathcal{U}}$. For instance, if p wants to live somewhere with a warm climate, whereas q wants to live in a metropolitan area, then they would (probably) have different options for where they study. Supposing that the universities in warm areas correspond to an open set $W \in \tau$, and those in metropolitan areas to an open set $M \in \tau$, then we compare the values of $\mathcal{U}(d, G^{-1}(W))$ and $\mathcal{U}(d, G^{-1}(M))$. If

$$\mathrm{Hom}_{\mathbf{FinOrd}}\Big(\mathcal{U}\big(d, G^{-1}(W)\big), \mathcal{U}\big(d, G^{-1}(M)\big)\Big)$$

is non-empty, then q has more options for d professors to study with than does p. Vice versa if

$$\mathrm{Hom}_{\mathbf{FinOrd}}\Big(\mathcal{U}\big(d, G^{-1}(M)\big), \mathcal{U}\big(d, G^{-1}(W)\big)\Big)$$

is non-empty.

4.4 Super Domains as Compositions

Summary. A method for translating super domains into elementary compositions is presented. This requires us first to introduce the notion of a *universal super domain.* Once this latter concept is defined, we will then be able to present the category of super domains.

— § —

In this section we demonstrate how a super domain $\mathscr{C}(\Sigma)$ can be translated into an elementary composition. Such a translation is highly fruitful, since it allows us to massively enrich the notion of a super domain, as well as the aggregates that it contains. Once the translation of super domains into elementary compositions is achieved, we no longer are required to keep aggregates $A \in \Sigma$ static, and can instead let them vary. We call such aggregates *variable aggregates*, and the super domain containing variable aggregates is called a *variable super domain*.

We will first define what we call a *universal super domain* Υ over a basis $\mathbf{B}$, and then show that a super domain over $\mathbf{B}$ corresponds to a diagram $\mathcal{D} : J \to \mathscr{C}(\Upsilon)$. When we convert J and $\mathscr{C}(\Upsilon)$ into structures, then such a diagram corresponds to an elementary composition, where J is the domain and $\mathscr{C}(\Upsilon)$ the codomain of the composition's content.

4.4.1 Universal Super Domains

Summary. The concept of a *universal super domain* is presented. A universal super domain is a super domain consisting of all possible aggregates over a basis. We then translate such constructions into constructions in **Str**.

— § —

Our first task is to define a super domain Σ in terms of a universal super domain Υ. The universal super domain over $\mathbf{B}$ is the super domain consisting of *all* possible aggregates with basis $\mathbf{B}$. Therefore for every $B \subseteq \mathbf{B}$, and for every $\mathfrak{mul} \in \mathrm{Hom}_{\mathcal{P}\mathbf{B}+\mathbb{N}^+}(B, \mathbb{N}^+)$, we have $(B, \mathfrak{mul}) \in \Upsilon$.

Now let Σ be a super domain over $\mathbf{B}$ and J an index category isomorphic to $\mathscr{C}(\Sigma)$. Then we can identify $\mathscr{C}(\Sigma)$ as a diagram

$$\mathcal{D}_\Sigma : J \to \mathscr{C}(\Upsilon). \tag{4.18}$$

Any super domain over $\mathbf{B}$ can be identified as a J-shaped diagram in $\mathscr{C}(\Upsilon)$, for J an arbitrary index category. Therefore our translation of a super domain Σ into an elementary composition requires a translation of J and $\mathscr{C}(\Upsilon)$ into structures, and the diagram $\mathcal{D}_\Sigma$ into a structure morphism.

Let us now provide this translation. We will see that J gets translated into a partial order and $\mathscr{C}(\Upsilon)$ to a partially ordered $\mathbb{N}$-module.[10] The functor $\mathcal{D}_\Sigma$ gets translated into a structure morphism satisfying some special conditions, to be discussed later.

[10]Note that in this section, the way that we conceive of a module is more general than the usual mathematical situation. In the latter context, a module consists of a group M along with an action by a ring R on M. In the current context, we let both M and R be more general kinds of objects. Specifically, we define an $\mathbb{N}$-module as a monoid $\mathbb{N}^n$ equipped with an action by $\mathbb{N}$ on $\mathbb{N}^n$.

(I) *Translation of J Into a Structure.* J corresponds to the underlying quiver of $\mathscr{C}(\Sigma)$. Since the morphisms in $\mathscr{C}(\Sigma)$ determine a partial order on Σ, we need merely to define the partial order relation $\preceq$ on the set J_0 of objects from J. So we can translate J into an elementary structure $(J_0, \{\preceq\})$, and thus we derive the functor $\mathrm{PSh}(\{\preceq\}) = Fu$ via Equation 3, giving us the structure

$$J_{\preceq} \xrightarrow[Fu \rightarrowtail @J_0]{} \mathbf{Simple}(@J_0).$$

(II) *Translation of $\mathscr{C}(\Upsilon)$ Into a Structure.* We first want to translate Υ into a structure corresponding to an $\mathbb{N}$-module whose basis vectors correspond to elements $b \in \mathbf{B}$. Then we need to define a partial order on this $\mathbb{N}$-module to give it the categorical structure of $\mathscr{C}(\Upsilon)$.

First, we need to show that Υ does indeed correspond to an $\mathbb{N}$-module. For $card(\mathbf{B}) = n$, we need to show that Υ corresponds to $\mathbb{N}^n$. Let's see how this is the case. We have for a subset $B \subseteq \mathbf{B}$ all the aggregates $A \in \Upsilon$ with support B. Therefore for every $\mathfrak{mul} \in \mathrm{Hom}_{mathcalP(B)+\mathbb{N}^+}(X, \mathbb{N}^+)$ we have $(B, \mathfrak{mul}) \in \Upsilon$. Denote by Υ_B the set of all aggregates in Υ with support B. This furnishes an injection

$$i_B : \Upsilon_B \rightarrowtail \mathbb{N}^n$$

that sends each $A \in \Upsilon_B$ to the element in $\mathbb{N}^n$ whose b-components are the multiplicity of b in A, with $b = 0$ when $b \notin B$. Note that if A is the (necessarily unique) aggregate whose support is the empty set, then $i_\varnothing(A) = (0, \ldots, 0)$, i.e., A gets mapped to the zero vector. We then get

$$\coprod_{B \in \mathcal{P}(\mathbf{B})} i_B(\Upsilon_B) = \mathbb{N}^n,$$

establishing the correlation between Υ and $\mathbb{N}^n$.

Now we define the module structure on $\mathbb{N}^n$. We showed how to define monoids as well as modules as elementary structures in Example 2.3. The only new addition we make here is to endow $\mathbb{N}^n$ with basis vectors corresponding to each $b \in \mathbf{B}$. We can specify these basis vectors by defining the following generator, which is an injection

$$\beta : \mathbf{B} \to \mathbb{N}^n$$

that associates each $b \in \mathbf{B}$ to its basis vector $v_b \in \mathbb{N}^n$. (While it is not totally necessary to define a module structure on $\mathbb{N}^n$, rather than e.g. a monoid structure, the module structure is a typical choice for constructing an object with 'coordinates'. In this context, a coordinate represents an aggregate over $\mathbf{B}$.)

We also partially order $\mathbb{N}^n$ to correspond to the partial order of $\mathscr{C}(\Upsilon)$. We define the generator $\leq : \mathbb{N}^n \to \mathbb{N}^n$ where for $u, v \in \mathbb{N}^n$, we have $u \leq v$ iff $u_i \leq v_i$ for all i ranging over the positions of u and v's vectors.

Thus for an elementary structure $(\mathbb{N}^n, R)$, we get $\mathrm{PSh}(R) = Gu$, and thus we can define the structure

$$\mathbb{N}^n_{\leq} \underset{Gu \rightarrowtail @\mathbb{N}^n}{\longrightarrow} \mathbf{Simple}(@\mathbb{N}^n),$$

which corresponds to $\mathscr{C}(\Upsilon)$. Specifically, $\mathbb{N}^n_{\leq}$ is a partially ordered $\mathbb{N}$-module, where the partial order corresponds to the morphisms in $\mathscr{C}(\Upsilon)$ and the elements of the module correspond to the objects in $\mathscr{C}(\Upsilon)$.

(III) *Translation of $\mathcal{D}_\Sigma : J \to \mathscr{C}(\Upsilon)$ Into a Structure Morphism.* Our last step is to translate the diagram $\mathcal{D}_\Sigma : J \to \mathscr{C}(\Upsilon)$ into a structure morphism $\Delta_\Sigma : J_{\preceq} \to \mathbb{N}^n_{\leq}$. This is simple, since the value of $\mathcal{D}_\Sigma$ on the objects of J is equivalent to the value of Δ_Σ on the elements of $J_{\preceq}$.

For partial orders P and Q, an order-preserving map $f : P \to Q$ is such that if $a \leq b$ for $a, b \in P$, then $f(a) \leq f(b)$. So let $\Theta \subseteq \mathrm{Hom}_{\mathbf{Str}}(J_{\preceq}, \mathbb{N}^n_{\leq})$ denote the set of all order-preserving structure morphisms between $J_{\preceq}$ and $\mathbb{N}^n_{\leq}$. We want to see if the set of J-shaped diagrams in $\mathscr{C}(\Upsilon)$ is in bijection with Θ, to ensure that each unique super domain $\mathcal{D}_\Sigma(J)$ corresponds to a unique order-preserving structure morphism $\Delta_\Sigma(J_{\preceq})$. In other words, we want to prove that sets $\mathrm{Hom}_{\mathbf{Cat}}\big(J, \mathscr{C}(\Upsilon)\big)$ and Θ are in bijection. We thus introduce the following

Lemma 4.1. *The set of order-preserving functions between partially ordered sets X and Y is in bijection with the set of functors between their corresponding posetal categories $\mathbf{P}X$ and $\mathbf{P}Y$.*

Proof. We prove this via the following sublemmas.

Lemma 4.4.1. *There is an injection from the set of functors from $\mathbf{P}X$ to $\mathbf{P}Y$ to the set of order-preserving functions from X to Y.*

Proof. Only one functor $F : \mathbf{P}X \to \mathbf{P}Y$ can correspond to the order-preserving function $f : X \to Y$. If there are functors $F, G : \mathbf{P}X \to \mathbf{P}Y$ such that $F(x) = G(x)$ for all $x \in X$, then this means that F and G correspond to the same order-preserving function. We show that if F and G are equal on objects, then $F = G$. If $F(x) = G(x)$ for all $x \in X$, then for any $x' \in X$ we of course have $\mathrm{Hom}_{\mathbf{P}Y}\big(F(x), F(x')\big) = \mathrm{Hom}_{\mathbf{P}Y}\big(G(x), G(x')\big)$. Since such hom-sets contain at most one element, we thus have that $F(i : x \to x') = G(i : x \to x')$ for all morphisms i in $\mathbf{P}X$. Hence $F = G$. Therefore there is an injection from the set of functors from $\mathbf{P}X$ to $\mathbf{P}Y$ to the set of order-preserving functions from X to Y. ☐

Lemma 4.4.2. *There is an injection from the set of order-preserving functions from X to Y to functors from $\mathbf{P}X$ to $\mathbf{P}Y$.*

Proof. Only one order-preserving function can correspond to the functor $F : \mathbf{P}X \to \mathbf{P}Y$, namely the function that is equal to F's value on objects. ☐

Since we have injections in both directions, namely from functors to order-preserving functions and vice versa, we therefore have a bijection. □

We thus get the following theorem.

Theorem 4.1. *The set Θ of order-preserving structure morphisms from $J_{\preceq}$ to $\mathbb{N}^n_{\leq}$ is in bijection with the set $\mathrm{Hom}_{\mathbf{Cat}}\big(J, \mathscr{C}(\Upsilon)\big)$ of functors from J to $\mathscr{C}(\Upsilon)$.*

Proof. Translate $J_{\preceq}$ and $\mathbb{N}^n_{\leq}$ into the posetal categories $\mathbf{P}J_{\preceq}$ and $\mathbf{P}\mathbb{N}^n_{\leq}$. From Lemma 4.1 we know that the set Θ is in bijection with $\mathrm{Hom}_{\mathbf{Cat}}(\mathbf{P}J_{\preceq}, \mathbf{P}\mathbb{N}^n_{\leq})$. Since we have isomorphisms $\mathbf{P}J_{\preceq} \cong J$ and $\mathbf{P}\mathbb{N}^n_{\leq} \cong \mathscr{C}(\Upsilon)$, we get the obvious bijection

$$\mathrm{Hom}_{\mathbf{Cat}}(\mathbf{P}J_{\preceq}, \mathbf{P}\mathbb{N}^n_{\leq}) \cong \mathrm{Hom}_{\mathbf{Cat}}\big(J, \mathscr{C}(\Upsilon)\big). \tag{4.19}$$

□

4.4.2 Super Domains as Elementary Compositions and Categories of Super Domains

Summary. We demonstrate how to translate super domains into elementary compositions. This will enable us to present the category of super domains.

— § —

For structures $J_{\preceq}$ and $\mathbb{N}^n_{\leq}$ where $J_{\preceq}$ is a partial order and $\mathbb{N}^n_{\leq}$ a partially ordered $\mathbb{N}$-module over the basis $\mathbf{B}$, a super domain over $\mathbf{B}$ can be encoded as an elementary composition

$$\mathit{SupDom} : \mathbf{Determinate}\big(J_{\preceq} \xrightarrow{\Delta} \mathbb{N}^n_{\leq}\big), \tag{4.20}$$

where Δ is an order-preserving structure morphism.

In the next few sections, we investigate categories of super domains.

4.4.2.1 The Category of Super Domains

Summary. The category of super domains is presented.

— § —

Let *PoSet* be the set of simple structures in **Str** that constitute partial orders on countable[11] sets. This means that each $P \in \mathit{PoSet}$ corresponds to an elementary structure $S = (X, \{\leq\})$, where $\leq$ partially orders X. Next, let *PoMod* denote the set of structures in **Str** that determine posetal $\mathbb{N}$-modules over arbitrary bases of compositions.

We have the category **SupDoms** as follows:

[11] A countable set is a set whose cardinality is less than or equal to the cardinality of $\mathbb{N}$.

1. For objects, we have the disjoint union $PoSet \sqcup PoMod$.

2. For morphisms, we have order-preserving structure morphisms $\theta : P \to X$ under the condition that $P \in PoSet$. For morphisms whose domain is an $M \in PoMod$, we have only the identity morphism Id_M. (We will explain later why we restrict morphisms on $\mathbb{N}$-modules to be identities.)

Now we want to define the category consisting of all super domains over some basis $\mathbf{B}$. Thus, for $M \in PoMod$ a module over the basis $\mathbf{B}$, take the over category $\mathbf{SupDoms}/M$, consisting of the following:

1. For objects, all morphisms $(\theta : P \to M) \in \mathbf{SupDoms}_1$ with codomain M.

2. For morphisms, all morphisms $(f : P \to P') \in \mathbf{SupDoms}_1$ such

$$\begin{tikzcd} P \arrow[rr, "f"] \arrow[dr, "\theta"'] & & P' \arrow[dl, "\theta'"] \\ & M & \end{tikzcd}$$

commutes.

Since each object $(\theta : P \to M)$ in $\mathbf{SupDoms}/M$ is an order-preserving structure morphism from a poset to an $\mathbb{N}$-module over $\mathbf{B}$, this means that an object in $\mathbf{SupDoms}/M$ corresponds to the content of an elementary composition, e.g. as in

$$SupDom : \mathbf{Determinate}\big(P \xrightarrow{\theta} M\big).$$

Therefore, we can think of the category $\mathbf{SupDoms}/M$ as constituting the category of super domains over $\mathbf{B}$. A morphism in $\mathbf{SupDoms}/M$ thus corresponds to a morphism of super domains.

4.4.2.2 Determining Morphisms Between $\mathbb{N}$-Modules

Summary. The problem of determining morphisms between $\mathbb{N}$-modules is discussed. The situation is not so obvious, as we will see.

— § —

The reason we restricted morphisms in **SupDoms** on $\mathbb{N}$-modules to be identities is (1) because we have no reason to map from modules to posets, and (2) because it is not clear what types of morphisms we should allow between modules. An obvious choice would be to allow for maps $\varphi : M \to N$ between modules that correspond to homomorphisms on their monoids. The reason that this does not suffice is for the following possibility. Suppose we define a super domain Σ over $\mathbf{B}$. We may realize that each $A \in \Sigma$ has some 'hidden' composition $h \notin \mathbf{B}$, so that when we go back to observe Σ we realize that each $A \in \Sigma$ has some h material. Thus we'd derive a new super domain Σ_h over $\mathbf{B} \cup \{h\}$. In such a situation, it may be the case that an aggregate A that precedes A' in $\mathscr{C}(\Sigma)$. However, once we add the hidden h components, we may derive an A_h and A'_h such that A_h does not precede A'_h in $\mathscr{C}(\Sigma_h)$. To see how such a situation can destroy the condition for a monoid homomorphism, consider a super domain Σ over $\mathbf{B}$ containing the null aggregate $A_\varnothing$. Let M_Σ denote module over $\mathbf{B}$. The null aggregate $A_\varnothing$ corresponds to the identity element (zero vector) in M_Σ. Now it may be the case that, later on, we observe a hidden basis composition $h \notin \mathbf{B}$ that is indeed present in many aggregates $A \in \Sigma$, so that we must extend $\mathbf{B}$ to $\mathbf{B} \cup \{h\}$ and Σ to Σ_h. If $A_\varnothing$ is one such aggregate that contains h, then a mapping $f : M_\Sigma \to M_{\Sigma_h}$ would send $0 \in M_\Sigma$ to a non-zero element in M_{Σ_h}. Since an $\mathbb{N}$-module homomorphism must map identity elements to identity elements, f is *not* such a homomorphism.

Another problem is that introducing the basis composition $h \in \mathbf{B}$ contained in many $A \in \Sigma_h$ could destroy the partial order on Σ. For instance, suppose we have $A_\varnothing, A_b \in \Sigma$, where $A_\varnothing$ is the null aggregate and A_b the aggregate consisting of the single basis composition $b \in \mathbf{B}$ with multiplicity 1. Then $A_\varnothing < A_b$. However, if, after realizing the existence of h in $A_\varnothing$ and A_b, we could find that $A_\varnothing$ contains more h compositions than A_b. For instance, let $0, v_b \in M_\Sigma$ denote $A_\varnothing, A_b$, respectively. Then under the desired mapping $f : M_\Sigma \to M_{\Sigma_h}$ that would send 0 and v_b to the aggregates that also contain their h components, we have $f(0) \not\leq f(v_b)$, and thus f is not order-preserving. Therefore the triangle

$$\begin{tikzcd}
 & P \arrow[dl, "\theta"'] \arrow[dr, dashed, "f \circ \theta \notin \mathbf{SupDoms}_1"] & \\
M_\Sigma \arrow[rr, "f"'] & & M_{\Sigma_h}
\end{tikzcd}$$

does not commute.

Because of such dilemmas, there is no way to introduce the desired morphisms between modules in **SupDoms**. However, if we remove all of the partial order information of the modules, we can get more flexibility. The price we have to pay is that we will no longer be able to encode the subsumptive hierarchy between aggregates in a super domain, since no order information will be given. However, what we lose in subsumptive information we gain in navigability.

We thus introduce the category **SupDoms***. This category is like **SupDoms** except less rigid. Instead of being built from the sets of structures $PoSet$ and $PoMod$, it is built from the sets of structures Set and Mod. The set of structures Set consists of all structures corresponding to sets. This essentially carries over the underlying set of a form into $\mathbf{Rel}^@$.

The set of structures Mod is like $PoMod$ just without the partial ordering relation defined on each structure in $PoMod$. We thus have the category **SupDoms*** consisting of the following:

1. For objects, we have the disjoint union $Set \sqcup Mod$.
2. For morphisms, we have functional structure morphisms $f : X \to Y$ under the condition that if $X \in Mod$ then $Y \in Mod$. All other functional structure morphisms are permitted.

We do not allow morphisms $g : M \to X$ where $M \in Mod$ and $X \in Set$ since we have no reason to map from modules to sets.

In **SupDoms***, a morphism $\Sigma : X \to M$ for $X \in Set$ and $M \in Mod$ can be called a *super domain morphism*, since it picks out a set of aggregates from the universe of all aggregates M. Since neither X nor M are partially ordered, there is thus no longer any encoding of the subsumptive relations – i.e., relations of inclusion – between aggregates. However, we can now navigate between super domain more freely. To use an earlier example from this section, let $M, M_h \in Mod$ be the universal super domains over bases $\mathbf{B}$ and $\mathbf{B} \cup \{h\}$, respectively. If we have a super domain morphism $\Sigma : X \to M$, and realize later that the composition $\Sigma(X)$ actually contains some hidden h compositions, then we can define a map $f : M \to M_h$ that makes

$$\begin{tikzcd}
 & X \arrow[dl, "\Sigma"'] \arrow[dr, "\Sigma'"] & \\
M \arrow[rr, "f"'] & & M_h
\end{tikzcd}$$

commute.

Thus f carries the aggregates in $\Sigma(X)$ to their corresponding aggregates in M_h. Therefore **SupDoms*** allows us to navigate between super domains over different bases, which was not achievable in **SupDoms**.

4.5 Variable Aggregates

Summary. Until now, aggregates have been presented as 'static' objects in the sense that their supports and multiplicities are fixed. However, there are many practical situations in which an object can gain and lose matter while still preserving its identity. Thus we present the concept of a *variable aggregate*, which is an aggregate whose support and multiplicity can vary.

— § —

As we saw in Example 4.4, an aggregate can vary in terms of its lower level compositional makeup. For instance, an oxygen atom can lose or gain protons, electrons, and neutrons. Nonetheless, we still refer to such an atom as an oxygen atom, regardless of such variations in its material constitution. We should therefore allow for a super domain to consist not just of static, but mobile aggregates as well. In this section we will define variable aggregates as well as variable super domains. But first, we will need to define a certain class of compositions that can, for now, be called *processes*.

4.5.1 Processes

Summary. The concept of a *process* is presented. Processes correspond to abstract networks that are realized concretely.

— § —

Processes constitute a class of compositions. The idea is that a process consists of a set of states, along with transitions between states. Thus a process constitutes some kind of directed graph. To encode such a process as a composition, we want the domain of the composition's content to be a directed graph, and we want its codomain to be some kind of 'state space'. This way, the content corresponds to an abstract graph being realized in a concrete way.

We can define a directed graph as we defined quivers in Section 2.4.2. Thus for a vertex set V, let

$$aDigraph \underset{Fu\rightarrowtail @V}{\longrightarrow} \mathbf{Simple}(@V)$$

denote a directed graph.

For S an arbitrary structure, we can then define a process as a composition

$$aProcess : \mathbf{Determinate}\big(aDigraph \xrightarrow{P} S\big).$$

4.5.2 Super Domains of Variable Aggregates

Summary. Now that we have presented variable aggregates, we should therefore consider the possibility of having super domains of variable aggregates. Such super domains are presented in this section.

— § —

To define a variable super domain, we first need to introduce variable aggregates.

Let $P \in PoSet$, $M \in PoMod$ and let $\Sigma : P \to M$ specify a super domain.

Next, let

$$\Gamma \underset{Fu\rightarrowtail @E}{\longrightarrow} \mathbf{Simple}(@E)$$

be a directed graph.

We first define a *pre-variable aggregate* as an elementary composition

$$PVA : \mathbf{Determinate}\big(\Gamma \xrightarrow{\pi} P\big). \tag{4.21}$$

Such a composition simply specifies a process of possible transitions between positions in P.

Now suppose we have a family $\{PVA_i\}_{i \in I}$ of pre-variable aggregates, each defined by

$$PVA_i : \mathbf{Determinate}\big(\Gamma_i \xrightarrow{\pi_i} P\big).$$

Given such a family of pre-variable aggregates, we want to derive variable aggregates, and then a variable super domain. We first translate each PVA_i into a structure in the canonical way:

$$Str(PVA_i) \xrightarrow[\mathrm{Id}]{} \mathbf{Limit}(\Gamma_i \xrightarrow{\pi_i} P).$$

We then get a variable aggregate as a composition

$$VA_i : \mathbf{Determinate}\big(Str(PVA_i) \xrightarrow{\alpha_i} M\big), \tag{4.22}$$

which corresponds to the map $\Sigma \circ \pi_i : \Gamma_i \to M$.

To get an entire super domain of variable aggregates, we can take two approaches. The first approach constitutes a variable super domain as a structure, the second approach as a composition. For the first approach, we define a variable super domain by taking the coproduct of all variable aggregates, as so:

$$VarSupDom \xrightarrow[\mathrm{Id}]{} \mathbf{Colimit}(VA_1, \ldots, VA_n). \tag{4.23}$$

For the second approach, we first define the coproduct structure

$$PreVarAggs \xrightarrow[\mathrm{Id}]{} \mathbf{Colimit}(Str(PVA_1), \ldots, Str(PVA_n)),$$

of pre-variable aggregates. We then construct the elementary composition

$$VarSupDom : \mathbf{Determinate}\big(PreVarAggs \xrightarrow{\Sigma^*} M\big). \tag{4.24}$$

The structure morphism Σ^* sends each pair $(\gamma, p) \in PreVarAggs$ to its corresponding aggregate in M. For instance, for $\gamma \in \Gamma_i$, we have $\Sigma^*(\gamma, p) = \Sigma \circ \pi_i(\gamma)$. See Figure 4.3 for a visual representation of a variable super domain.

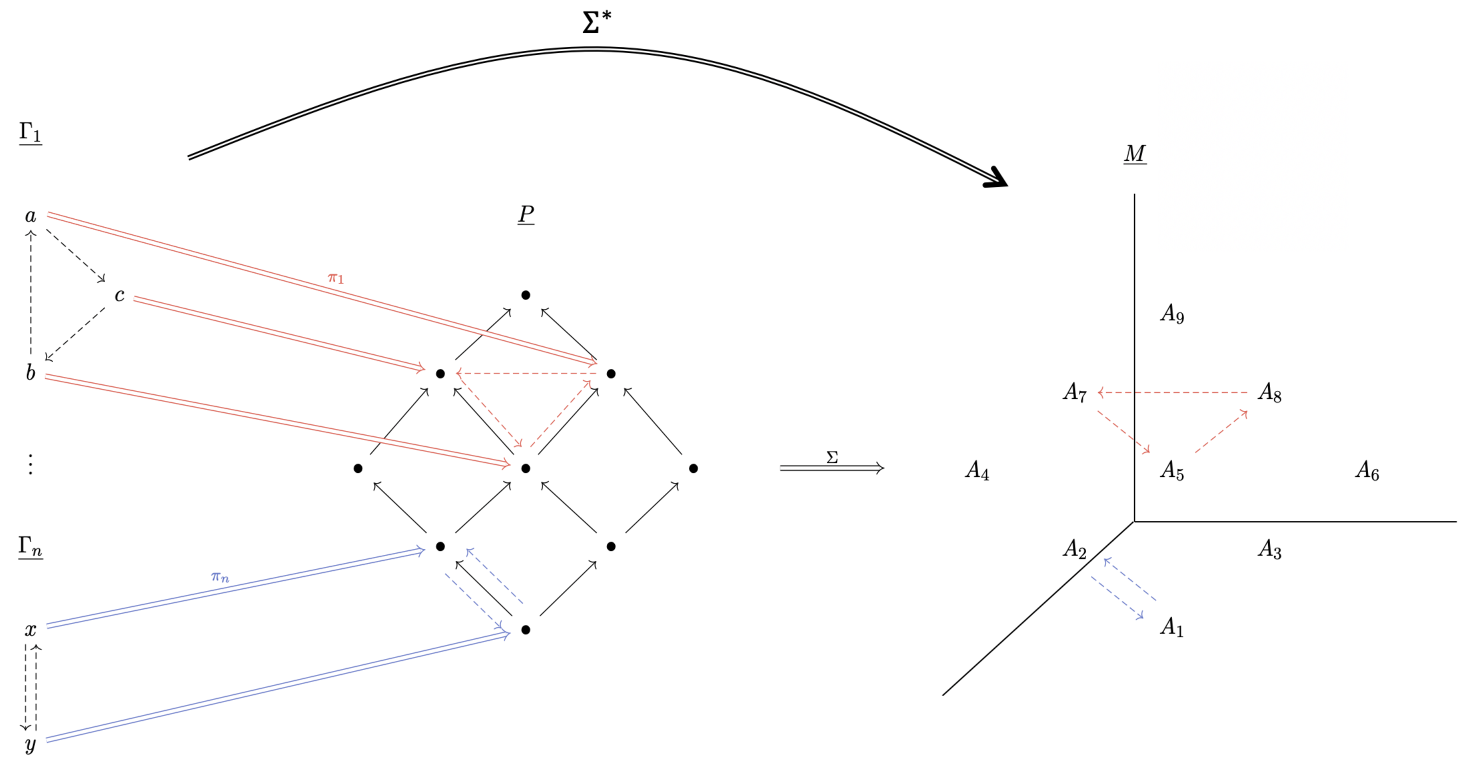


Figure 4.3: The abstract graphs $\Gamma_1, \ldots \Gamma_n$ are mapped into P, and the latter is then mapped into M, which contains aggregates. Composing the morphisms $\Sigma \circ \pi_i(\Gamma_i)$ establishes the process by which an aggregate can change its material constitution, while still preserving its identity.

4.6 Aggregates with Real-Valued Multiplicities

SUMMARY. We close the chapter by considering the situation where aggregates have real-valued multiplicities, rather than the natural-valued multiplicities presented in this chapter.

— § —

To conclude this chapter, we consider the situation where aggregates take multiplicities from $\mathbb{R}$ rather than $\mathbb{N}$. There are many contexts in which this will be the better formalism, some of which are the following:

- **Chemical Mixtures.** Suppose someone is constructing aggregates from certain chemicals like water (H_2O), sulfuric acid (H_2SO_4), and the like. In such a context, it would be ridiculous to measure an amount of water based on how many H_2O molecules it contains. Instead, one would use some other metric, like mass, or some measuring value like tablespoons, etc. In this situation, a super domain Σ over a basis $Chem$ of some chemicals should have for its multiplicities values in $\mathbb{R}$.

- **Paintings.** One might want to analyze a painting P as a compound composition over a basis $Colors$ of colors of paint. In this case, it doesn't make sense to consider P in terms of an aggregate that assigns quantities from $\mathbb{N}$ to each $c \in Colors$. Instead, it would be more useful to conceive of some metric such as 'surface area', and assign each $c \in Colors$ to the $x \in \mathbb{R}$ that is equal to the surface area that c takes up in the painting.

- **Sonic Spectrograms.** A sonic spectrogram displays the frequency and amplitude information of a given sound. For each frequency, a certain amplitude value is given. Since amplitude values are taken from real numbers, we can define a 'snapshot' of a sonic spectrogram as an aggregate whose support consists of all frequencies in the snapshot, and whose multiplicities are the amplitudes of the frequencies.

One can imagine many other examples of super domains whose aggregates should have real-valued multiplicities. For the most part, the formalism for dealing with real-valued aggregates is the same as the formalisms of this chapter dealing with natural-valued aggregates.

There is a crucial difference however between aggregates with $\mathbb{N}$-valued multiplicities and those with $\mathbb{R}$-valued multiplicities. In the former case we are using discrete quantities, in the latter continuous. Discrete quantities are in a sense absolute. For instance, if there are 10 objects of kind x, then no matter the context, there will still be this collection of 10 x's. On the other hand, real-valued quantities depend on a choice of metric, e.g. mass, weight, surface area, etc. So the situation where aggregates have real-valued multiplicities requires a specification of the metric used to evaluate the quantities of basis substances. It is important that this metric remain fixed for each aggregate of the super domain, or else the relations between aggregates in terms of their multiplicities will be unclear.

As a final note, we emphasize a fundamental fuzziness with respect to real-valued aggregates. To actually identify a real number would require infinite precision, since a real number x has infinite values after the decimal. When we refer to real numbers, for instance in scientific measurements, we necessarily must round off. Thus a real-valued measurement is implicitly an open set U of real numbers, where for $x, y \in U$ our tools are not precise enough to distinguish between x and y. Therefore real-valued aggregates will, in any practical circumstances, have margins of error.

Part II

Logic and Worlds

Chapter 5

On Meaning

5.1 Representationalism and Inferentialism

SUMMARY. We discuss two paradigms for formulating what constitutes meaning – namely *representationalism* and *inferentialism* – and discuss their respective features.

— § —

5.1.1 Representationalism

SUMMARY. We present an overview of how the concept of meaning is treated according to a representationalist perspective.[1]

— § —

The question of what constitutes meaning has a rich history. Our task in this chapter is not to provide a historical overview of the concept of meaning, but instead to elaborate on two philosophical positions, namely, *representationalism* and *inferentialism.* Our objective is not to side with one or the other, only to bring to the reader's attention to two compelling viewpoints that provide context for the rest of this part of the book.

The *representationalist* theory of meaning is the classical one; it contends that "to *mean something* is to *stand for something*"[2]. For instance, when I refer to a pen by calling it 'this pen', then the meaning of 'this pen' is the object itself, namely the pen. Dealing with properties is more subtle. When I utter "This pen is red" then there is (1) the meaning of the term 'this pen', and (2) the meaning of the property of redness. While the meaning of the term 'this pen' is a concrete object, the meaning of 'redness' is not a concrete item, but an abstraction (a universal); specifically, according to representationalism, properties

[1] For a historical overview, see [18].

[2] [18, p. 1084]

denote *classes* of items, and therefore 'redness' is interpreted as the class of items that are red. Therefore, the statement "This pen is red" is given semantic content by checking whether or not the object referred to by 'this pen' is in the class of items that have the property of being red.

Representationalism is thus built on the distinction between syntax and semantics, which correspond, respectively, to internal and external reality. Syntax consists in rules for constructing grammatically correct sentences, as well as drawing valid inferences from those sentences. An inference rule consists of a legitimate way to derive one statement from another statement. The most famous inference rule is called *modus ponens*, which says that given the statement "If P is true then Q is true", and the verification that P is true, one can therefore conclude that Q is true. Logically, this can be notated in sequent form as

$$P \implies Q, P \vdash Q.$$

On the other hand, semantics is the 'external reality' that verifies the truth of syntactically correct statements. The classical version of this idea as conceived by Tarski is, roughly, the following. We have a set[3] D of individuals that we wish to reason about. The symbols in the language that are meant to refer to the individuals in D are called *constant* symbols. We also have property symbols $P_1, \dots, P_n$ that denote properties that can hold of these individuals, as well as n-place relations that can hold between them[4]. Then, for each property/relation symbol P of arity k, there is a subset $\mathfrak{P} \subseteq D^k$, the elements of which are k-tuples for which the property/relation denoted by P holds. For instance, if we have the relation symbol $\leq$ and the domain $D = \mathbb{N}$, then the relation denoted by $\leq$, which we can notate as $\mathfrak{lt}$, is the subset of $\mathbb{N} \times \mathbb{N}$ that consists of pairs $(\mathfrak{i}, \mathfrak{j})$ such that $\mathfrak{i}$ is less than $\mathfrak{j}$. More precisely, the *statement* "$i \leq j$" is true iff $(\mathfrak{i}, \mathfrak{j}) \in \mathfrak{lt}$. In general, for a relation symbol R and constant symbols x, y, the statement xRy is true iff the objects denoted by x and y stand in the relation denoted by R. This is kind of funny, when one realizes that this is like saying that "This pen is red" is true iff this pen is red.

The representationalist paradigm proceeds from the bottom-up. We first have a domain D of objects, and then n-place atomic[5] predicates that classify those objects. This constitutes a sort of 'ground zero' of predicates. The logical connectives $\wedge, \vee, \implies$ as well as the quantifiers $\exists, \forall$ act as tools for synthesizing compound predicates. For instance, given the 1-place atomic predicates P and Q, we can form the compound predicate $P \wedge Q$. A pair of objects x, y satisfy $P \wedge Q$ when $P(x)$ is true and $Q(y)$ is true. This therefore determines a subset $\mathfrak{P} \subseteq D \times D$. We can keep building up more complex predicates by stringing them together; e.g. we can create the predicate $(P \wedge Q) \vee R$, again determining a subset of some k-fold product of D. It is therefore clear that *the meaning of a compound*

[3]This set is called the 'domain of discourse', or simply 'domain', for short.

[4]Note that a property is simply a 1-place relation.

[5]Atomic predicates are those of the form $P(x_1, \dots, x_k)$, i.e., those without any logical connectives and/or quantifiers.

predicate is determined by the meaning of the atomic predicates that constitute it. This is how, in the representationalist paradigm, meaning is generated from the ground-up. We start with atomic predicates, and with the use of synthetic tools – viz. connectives and quantifiers – we compose them together to form compound predicates.

Regardless of one's ideological preference for a representational or inferential theory of meaning, there's no doubt that the Tarskian framework furnishes a powerful computational resource.

5.1.2 Inferentialism

Summary. We present an overview of inferentialism.

— § —

A contrasting view of meaning formulated in large part by Sellars in [19, 20, 21] and Brandom in [4] is called *inferentialism.* Inferentialism also shares a resemblance to the later Wittgenstein's [22] theory of meaning-as-*use.* According to the inferentialist theory of meaning, a statement does not get its semantics by virtue of its correspondence with an external reality. Instead, the meaning of a statement is determined via its inferential role in a web of other statements. For instance, when I make a claim such as "x is a dog", then I am committed to also claim that "x is an animal". But I am also committed to answer other questions pertaining to e.g. properties that a dog may have, such as whether it has fur or hair, what color it is, its breed, etc. So, the meaning of "x is a dog" is *produced* via such a process of unpacking. In this situation, there is no external reality that contains the concept "dog", along with an individual x, that allows one to confirm whether or not x satisfies the property of being a dog. Instead, meaning is produced via the dialectical process of *giving and asking for reasons*[6], which is a process of elaborating upon the connections that an utterance has to other statements that justify it. In other words, "meaning is not a thing stood for by an expression", but rather "a *role* the expression assumes vis-à-vis the rules that govern it."[7]

Unlike the Tarskian logical apparatus for dealing with semantics, which is well-developed and now has diverse applications, there is minimal work in logic coming from an inferentialist perspective. Something that is similar to the inferentialist framework in logic is the work done in formalizing game semantics. In this context, the objective of the game is to prove (for the assertor) or disprove (for the refuter) a given proposition. However, the semantics is still formalized according to the Tarskian framework, which still upholds a distinction between the language-game and the external reality that acts as a 'referee'. However, there is an area of logic developed by Jean-Yves Girard's, called *Ludics*[8], which provides a logical framework that captures the crux of the inferentialist argument. In this work, there

[6] [4]

[7] [18, p. 1092]

[8] [8]. A good introduction to *Ludics* is given in [12].

is finally a logical formalism that (apparently) provides the means to meaning-production that is independent of the syntax-semantics distinction.

In contrast to the representationalist paradigm, the inferentialist paradigm proceeds from the top-down. Given two language-users, You and Me, You utters some proposition P. If P consists of subpropositions $A, B, \ldots$, then You must be prepared to justify them when challenged by Me. Suppose Me chooses to focus on the subproposition A. Then You must be prepared to justify A, which Me will then challenge by asking You to justify A's subpropositions $A_1, A_2, \ldots$, and so on. Meaning is thus generated by iteratively unpacking a more complex proposition into its simpler components. Moreover, this iterative unpacking is not necessarily deterministic, so meaning is 'context-sensitive', insofar as the meaning of a proposition P depends on the specific dialogue that follows the utterance of P. Meaning is also not determined by an independent reality, but generated immanently to the dialogue itself, and therefore the inferentialist paradigm is much more dynamic than the representationalist one. (In fact, the representationalist paradigm cannot *itself* support dynamism. If one wishes to incorporate dynamic predicates into a representationalist paradigm, he must do so via more 'artificial' devices, for instance by injecting a *temporal parameter* into the predicate language, as is done to define e.g. variable sets. But even then the temporal parameter *itself* remains static!)

Summarizing the difference between representationalism and inferentialism, we may say that in the representationalist paradigm, *compound* meaning is produced, whereas in the inferentialist paradigm, *foundational* meaning is produced.

5.2 Uses and Abuses

Summary. We discuss the uses and abuses of both the representationalist and inferentialist paradigms.

— § —

It seems that inferentialism provides a more convincing depiction of the *actual* processes that take place when meaning is being generated. It provides the machinery for investigating how we start with fuzzy concepts and eventually arrive at more precise concepts. It also provides an intuitive yet systematic understanding for something that happens all the time, namely a concept changing over time. No longer in the inferentialist paradigm do predicates denote static sets. Rather, the meaning of a predicate is contingent upon the context in which it is uttered, and furthermore a predicate whose meaning has been settled, e.g. in some dialogue, can be brought back into question and therefore revised, without such a revision implying a destruction of the original meaning. Even in disciplines where formal (Tarskian) predicates abound, such as the sciences, there is still the process of the *genesis* of such predicates, and their genesis is not adequately framed by the Tarskian perspective.

However, there are of course times when a more rigid database of predicates is required. This is when a representationalist paradigm is likely the practical choice. For instance, a predicate such as ‘is prime’ is clearly compliant with a Tarskian semantics. This is because in mathematics, the domains of discourse are – *in some sense* – independent of our formalisms of them. For instance, take two encodings of the natural numbers:

- Zermelo ordinals
    1. $0 := \varnothing$
    2. $1 := \{0\} = \{\varnothing\}$
    3. $2 := \{1\} = \{\{\varnothing\}\}$
    4. $3 := \{2\} = \{\{\{\varnothing\}\}\}$
    5. $\vdots$
- von Neumann ordinals
    1. $0 := \varnothing$
    2. $1 := \{0\} = \{\varnothing\}$
    3. $2 := \{0, 1\} = \{\varnothing, \{\varnothing\}\}$
    4. $3 := \{0, 1, 2\} = \{\varnothing, \{\varnothing, \{\varnothing\}\}\}$
    5. $\vdots$

It is clear that (other than for 0 and 1) when we put Zermelo ordinals in correspondence with von Neumann ordinals, they do not satisfy a set-isomorphism, since their sets for each ordinal are of different cardinalities. For instance, according to Zermelo’s method, the set representing each ordinal always has a cardinality of 1, whereas von Neumann’s method produces sets of a cardinality equal to the set’s respective ordinal. But this simply means that the *set-theoretic* structure of the two encodings is different; it does not mean that the *structure of the ultimate object* that both constructions encode (namely, the natural numbers) is different. To treat two structures as the same does not require that they be the same all the way down to their origins. All that is required is that they be the same with respect to the certain *aspect* of them that we choose to focus our attention upon. When we denote a mathematical object, such as $\mathbb{N}$, our denotation is conditioned by an implicit agreement that the mathematical community has with regards to the *specific* aspect of $\mathbb{N}$ that we are interested in. In the case of $\mathbb{N}$, the specific structure that we are interested in is, generally speaking, a totally ordered set of infinite size. When presented with an ordinal n, we rarely ask for its set-theoretic encoding.

Returning to our point on defining predicates on such mathematically precise domains of discourse, we see that predicates such as ‘is prime’ are invariant with respect to different encodings. For instance, we may have any encoding of $\mathbb{N}$ as we like, and such a choice will not affect the fact that 3 is a prime number.

Predicates such as 'is prime', along with their amenability to simple and precise verification, are the kinds of predicates that abound in mathematics. For instance, if we are given a set F of functions $f : X \to Y$, and X and Y are endowed with topological structure, we may want to know which of those functions are continuous. Here again we have a precise criterion for whether or not a function is continuous, and therefore we easily get a set $C \subseteq F$ of continuous functions, by verifying which elements $f \in F$ are indeed continuous as determined by the criterion of continuity.

Of course, predicates defined on rigidly defined domains of discourse are the exception, not the rule. A Tarskian paradigm for dealing with predicates falls apart when we are confronted with checking predicates that are more loaded with implications, such as e.g. whether or not a piece of music is 'beautiful', or a duck is 'annoying'.

We can crudely summarize the virtues of representationalism and inferentialism by saying that *representationalism is a better scientific instrument, whereas inferentialism is more correct with regards to the process of meaning-production.*

5.3 *Is* Versus *Ought*

Summary. In this section, we will see that the representationalist paradigm provides the framework for resolving what *is* the case. On the other hand, the inferentialist paradigm paradigm provides the machinery for investigating what *ought to be* the case.

— § —

5.3.1 The Question "What is x?" as an Implicit Normative Question

Summary. We suggest the thesis that questions of the form "What is x?" are questions involving what x *ought to be.*

— § —

There are two ways to view a question of the form "What is x?". The common view is that x has some implicit nature or essence, and that our answer to the question should reveal the essence of x. Or, even if we admit that there are no such things as 'essences', and that reasoning consists of mental representations of phenomena, as Kant would maintain[9], we often still assume that there is something like an essence inherent to our reasoning conventions (as Kant presumed). Therefore when we answer the question "What is x?", a Kantian would maintain that we are not revealing the nature of the *Ding an Sich*, but he

[9] [9]

may still maintain that we are revealing a sort of 'logical essence', such as the essence of our representational apparatus.

A contrasting view is that the question "What is x?" is the originary site of a process whereby x acquires a kind of 'deontic status'. The idea is that when we ask "What is x?", it is not that we are gradually revealing the essence of x. Rather, we are defining what x *ought to be*: for instance, what x *ought to be* in order for us to use it in the ways that we do. Try asking for yourself "What is x?" for some x, and see if your attempts to define it are not attempts to understand what x *ought to be* in order for you to carry out certain activities.

This view suggests that there is a world of action W in which x is embedded, and that attempts to define x are, in reality, attempts to firmly establish x's role in the world W. Such a process, whereby the meaning of x is gradually revealed by explicating x's role in W, is the process whereby x *attains deontic status.* The process by which objects acquire deontic status – which is through a process of *conceptualization* – is a driving force of human conduct. Through it, the world transforms from anarchy to order, by virtue of the fact that the concept's use becomes regulated.

Note, however, that the world W can become the object of the question "What is W?". Such a question requires a higher-order world $\mathcal{W}$ where W exists as an object, and itself attains deontic status via the process of explication of W's role in $\mathcal{W}$. This process of explication changes W itself, and therefore changes the objects x that occur in W. We can then continue the question, asking "What is $\mathcal{W}$?", where $\mathcal{W}$ occurs in $\mathfrak{W}$, and so on.

5.3.2 The Pragmatic Nature of Meaning-Production

SUMMARY. We discuss the pragmatic nature of meaning-production.

— § —

Let us return to the concept of meaning in the inferentialist paradigm. In this paradigm, meanings are not *a priori* given via some external semantics that verifies statements; rather, meanings are produced via dialectical elaboration upon a certain statement, generating threads of inferences. The inferentialist paradigm captures the idea that humans, as linguistic beings, communicate with one another by giving and asking for reasons. When you make a claim P, then your claim entitles me to ask you for reasons that justify your claim that P is the case. Upon receiving justifications from you, I can yet-again ask for justifications of those justifications, and so on. While this process of giving and asking for reasons can, in principle, continue indefinitely, it is perpetually being stopped for a specific reason: namely, because *humans create finite descriptions of things in order to specify what traits those things ought to have if they are to be used in a certain way.* In other words, human discourse has as one of its objects the project of creating *tools.* A *concept-as-tool* does not arise from an infinite process of dialectical refinement. Rather, it arises by suspending the discourse and setting as final certain propositions. We will see an example in section 5.4.

So, meaning is produced in order to establish rules of human conduct. For instance, consider the following dialogue between agents A and B:

> A: Here's a new object I made.
>
> B: What is it?
>
> A: It's a metal shaft with a large metal object at the top.
>
> B: What's it used for?
>
> A: Well, you can bang it against things with great force.
>
> B: Why would I want to do that?
>
> A: If you have some wood that you would like to connect together, you can grab some metal nails and bang a nail through one piece of wood to another, therefore connecting them both.
>
> B: I see. What do you call it?
>
> A: A *hammer*.

In this situation, the meaning of the object under consideration – the hammer – is generated via the dialogue between A and B. However, what meaning was generated? All that was really expressed was the fact that this new metal object can be used to bang certain things, such as nails, into other things. The full applicability of the hammer was not expressed, but only a partial description of what constitutes a 'proper' use of the object. For instance, if B has violent tendencies, then he may use the object to kill someone. In this case, we can say that the meaning of *hammer* was changed by B's extending its range of application. This is far from the Tarskian situation, where the concept 'hammer' would be conceived as a predicate that classifies all objects that satisfy some abstract criterion for being a hammer. In the Tarskian situation, there is no mention of the object's use, and furthermore the object is conceived of totally independently of its use. This makes the meaning of 'hammer' purely static. On the other hand, in the inferentialist framework, the meaning of 'hammer' is dependent on the context in which it is explained, and this seems very natural when it comes to the dynamic process of humans acquiring new concepts.

5.4 The Path to Formal Predicates

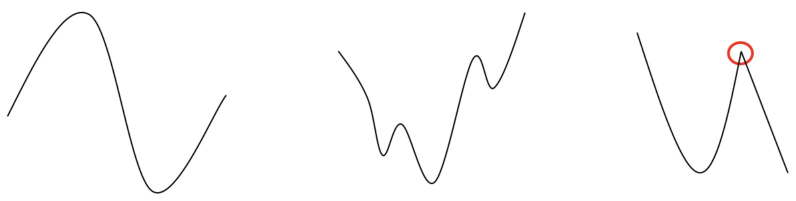
Summary. Our goal in this section is to show how *the possibility for representation is conditioned by inferential processes of meaning-production.* Two experiments are provided to articulate this notion.

— § —

Let us now demonstrate two experiments in meaning-production.

Experiment 5.1. Consider the following dialogue between A and B.

> A: This curve is smooth.
>
> B: What does it mean for a curve to be smooth?
>
> A: Here, look at these examples of curves. All of them are smooth, except for the one to the right which has jagged edge, circled in red.



> B: And so that means it's not smooth?
>
> A: Correct.
>
> B: Ok, but then how would I, when given any curve, be able to tell if it's smooth or not. Clearly it would be cumbersome if I had to look at the entire curve, and somehow notice a jagged edge. Also, couldn't there be other ways in which a curve could not be smooth, even if it doesn't have a jagged edge? Given your justifications for what constitutes a smooth curve, I am still not confident in what actually *is* a smooth curve.
>
> A: You are correct. Let's try to formulate this more precisely ...

This is the beginning of a dialogue that will eventuate in a fully rigorous definition of what constitutes a smooth curve. Looking at this debate from a historical perspective, it was the case that in order to define a curve, one had to define continuity; and in order to define continuity, one then had to construct point-set topology. Once the notion of a topological space was given precise axioms, we were then able to define a continuous function between such topological spaces. When we were able to understand what constitutes a continuous function, we were then able to define a curve as a continuous function $f : [0, 1] \to X$ from the unit interval into some topological space X. We also needed to define what it means for a function to have a derivative[10], and so on. Finally, we can then define a smooth curve as a continuous function $\gamma : [0, 1] \to \mathbb{R}^n$ that is continuously differentiable. Now, we may replace the entire dialogue above with the following one:

[10] Of course, the concept of 'derivative' was conceived before topology and therefore continuous functions etc.

A: This is a smooth curve.

B: What's a smooth curve?

A: A function $\gamma : [0,1] \to \mathbb{R}^n$ that is continuously differentiable.

Furthermore, the meaning of the statement "this is a smooth curve" no longer requires a dialogical process of explication. The meaning of "this is a smooth curve" is now completely encoded via the axioms of topology. So now 'smooth curve' is a Tarskian predicate, as it takes as its domain of discourse the set F of continuous functions, and is true for $f \in F$ whenever f is of the form $f : [0,1] \to \mathbb{R}^n$ and is continuously differentiable.

However, a crucial point is that *it was the dialogical process of giving and asking for reasons that led to the formal encoding of the concept of 'smooth curve'.* Therefore, the inferentialist paradigm provides the framework that allows us to see the process by which a fuzzy concept becomes totally unfuzzy. This is very much like how the function of Socratic dialogue is to move from a fuzzy concept to an Idea, an Idea being more like a Tarskian predicate. This leads us to the following

Dictum 5.1. *Reference is impossible without an inferential process that conditions it.*

The possibility of fully precise reference is an indication of how successful an inferential process was. For instance, the fact that I have a fully precise reference for the concept 'smooth curve' means that the historical-inferential process of defining what constitutes a smooth curve succeeded. Of course, the question of how successful an inferential process really is is by no means simple. This is why mathematical concepts are so-often revised. For instance, when Frege sought to answer the question "What is Number?", he came up with the formalism as outlined in [7], where numbers correspond to extensions of concepts. Someone could have asked him, "Is this object, x, the number 0?", and he could respond by saying "x is 0 iff x is the extension of the concept 'not self-identical'." However, upon learning of Russell's paradox, this logical framework for defining Number was shown to be insufficient, and therefore Number had to be redefined. This gives us a

Principle 5.1. *The reference of a concept can change when the rules of the game (of the inferential process of which the concept is the result) change.*

Experiment 5.2. Consider the following dialogue, again between A and B.

A: This rug is ugly.

B: How so?

A: The colors don't blend well.

B: I agree.

This dialogue has some crucial differences from the first dialogue between A and B. Notice that B requires A to justify the statement "This rug is ugly", which A does by stating that the colors don't blend well. At this point, B is in agreement with A that the rug is ugly (because the colors don't blend well). But *what* exactly are A and B agreeing on? In this situation, it is clear that there is no Tarskian predicate for 'ugly', insofar as neither A nor B has provided a necessary and sufficient condition for an item x to be ugly. We can say that they are both referring to the same *object* (the rug), but not that they are referring to the same *property* ('ugly').

Also, notice that B's first response is "How so?". This is a much different question than "What *is* ugliness?", which is the type of question that usually has the potential to evolve into a Tarskian predicate. We could say that 'ugly' is just a name, and that it may denote different predicates depending on the context. Thus, when I say "x is ugly", and you respond with "How so?", your response could be interpreted as the question "*Which* predicate bearing the name 'ugly'?", where there exists some set of predicates $\{P_1, \ldots, P_n\}$ all having the name 'ugly'. But still, this implies that there is a set of ready-made properties P_i that I am simply selecting from, which is not at all how discourse *actually* proceeds. At the end of the discourse, there is agreement on the meaning of the statement "This rug is ugly" even though the property 'ugly' remains fuzzy.

However, a thorough skeptic *would* take the dialogue further, as he would not be satisfied to come to agreement on a fuzzy proposition. Such is the skepticism of Socrates: by asking his interlocutor to define a certain concept, Socrates will only concede once his interlocutor has fully removed all fuzziness from the concept (– which never happens). For Socrates, fuzzy reality is Hell, and he is condemned to observe others unwittingly drag themselves into the tormenting labyrinths constituted by the use of such fuzzy concepts. It is thus the job of the Philosopher-King to save the concept from its abuse by the unenlightened, returning it to purity. In other words, the philosopher's job is to reveal the Tarskian predicates that underly fuzzy concepts. Let us take up this task in the sequel, but with a slight change of viewpoint: Instead of seeing the task *reductively* – as a process by which a hidden hyperuranion is *revealed* –, let us look at the task *productively* – as a process by which fuzzy concepts are *transformed* into Tarskian predicates. The ascent to the hyperuranion is real; it is just that Heaven is not given, but produced.

5.5 Warning!

SUMMARY. We close this chapter with a warning to systematic philosophers.

— § —

Let us first return to the problem in Section 5.3 of *Is* versus *Ought*. By *Is*, I mean the status of being fixed. This idea of *Is* is that which underlies the representationalist paradigm. When I can state that something *Is* the case (e.g. that an object *Is* a vector space), this means that I have a formal procedure for verifying whether that something is indeed the case. This requires that the domain of discourse (of the language that I'm using) remains *fixed*. As an example, consider mathematical objects. Once I construct the axioms for a vector space, then I no longer need to justify that some V is indeed a vector space. I simply need to check that the axioms are satisfied and then I'm done; at this point, I have verified that V is a vector space – no uncertainty involved. Vector spaces, Compositions, Tarskian predicates, and so on – all of these objects lay claim to a totally unambiguous (non-fuzzy) existence. But any formalism of what *Is* must be 'closed', to the extent that the formalism can only account for what *Is* according to its descriptive procedures. There are many *informal* reasons why this fact destroys the hope for a truly Universal System of concepts, but there is also the *formal* reason demonstrated by Gödel's incompleteness theorem, which shows that our formal systems are inherently limited in their ability to confirm what *Is* the case. The incompleteness theorem reveals a hard-wired limitation when it comes to formalizing what *Is*. We may thus say that every *Is* constitutes a *protocol*. A protocol consists of a (finite) set of fixed procedures that determine a realm of possibility. The realm of possibility determined by a protocol is by no means exhaustive of the realm of possibility *as such*.

However, we should not revolt at the idea of a protocol. There are good protocols and there are bad protocols: the good ones achieve what they set out to achieve, the bad ones don't. Every theorist knows what it's like to create a bad protocol, and that's why he revises his work. For example, Frege's *Grundgesetze* and ZFC set-theory are both attempts to establish (what are now called) *classical* foundations for mathematics. However, Frege's *Grundgesetze* fails due to logical inconsistency, whereas ZFC succeeds. Thus we can say that the *Grundgesetze* is a *bad* protocol whereas ZFC is a *good* protocol.

Once established, classical logic proved to be an immensely powerful tool. It aids greatly in our understanding of mathematical objects, and allows us to construct many wonderful things. It is both highly general and highly intelligible. Nonetheless, it would be too extreme to say that it solves the problem of reasoning once and for all. There are other phenomena that are better-captured by different logics, such as intuitionistic or substructural logics. It is not that classical logic *can't* describe the objects described by intuitionistic or substructural logics; it is simply that to describe such phenomena classically requires a tremendous amount of formal bloat, which affects the intelligibility of the phenomena-in-question. In fact, when confronted with such objects, classical logic behaves very much like a *bad protocol*: it creates a lot of bloat, and poorly executes the task at hand.

The dual notion of *Is* is *Ought*. By *Ought*, I mean the process of constructing a protocol according to specific *values*. When I ask myself "What *Is* x?", I am putting myself in the situation of having to determine what x *Ought* to be. Unlike the *Is* situation, where objects exist in a fixed manner, the *Ought* situation is where the nature of objects is in a state of flux. The Creative forces that mould Nature are at play when *Ought* is present, and

Ought is always present, as it is a real *process* of thought and existence. However, without arriving at an *Is*, the *Ought* is a *failure.* This gives a

Dictum 5.2. *Every* Ought *has as its telos an* Is*, that is, a protocol.*

Therefore it is the task of systematic philosophers to create protocols. To strive to create systems that are both highly *general* – so as to account for things and topics as broad in scope as those of philosophy –, yet highly *intelligible* – so as to be pragmatically efficient. However, due to the impossibility of having a final protocol, philosophers can never assume that they have realized a 'final construction' – there will always be something *outside* of their constructed domains of discourse. Thus the dual task of philosophers is to accept the inevitable demise of all worlds. If one has succeeded in building a general and intelligible system, then it is good to use it as long as it gives no issues. However, when such a system begins to act awkwardly when confronting a new kind of object, it is better to abandon it and devise a new system that is more appropriate to the object. The most important thing is to remain always thoughtful. This requires humility, yet also boldness. Boldness gives the courage needed to approach big problems, whereas humility is required to conform to their nature. An attitude which realizes a balance between boldness and humility is perhaps the following

Principle 5.2. *Take* Thinking *seriously, but not* Thoughts.

Chapter 6

Concepts

6.1 Explication

Summary. We outline the problem of *explication*, as defined by Carnap in [5, §I].

— § —

6.1.1 Carnap's Criterion

Summary. Carnap's concept of, and criterion for, explication is discussed.

— § —

In [5, p. 3], Carnap states that

> the task of *explication* consists in transforming a given more or less inexact concept into an exact one or, rather, in replacing the first by the second. We call the given concept (or the term used for it) the *explicandum*, and the exact concept proposed to take the place of the first the *explicatum*.

Although not a necessary requirement, an explication will often make use of an artificially constructed language, in order to facilitate a more exact description of the concept.

Carnap also provides four requirements that an explicatum must fulfill: (1) similarity to the explicandum, (2) exactness, (3) fruitfulness, (4) simplicity.[1] The idea is that prescientific concepts are transformed into scientific concepts via explication, which consists in more rigorously articulating the rules for the concept's use. An example given by Carnap is the concept Fish. The prescientific concept of fish is 'animal living in water'. According to this concept, whales and seals, for instance, are fish. An explication of Fish, referred to as Piscis by Carnap, is defined as 'lives in water, is a cold-blooded vertebrate, and has gills

[1] [5, p. 5].

throughout life'. Piscis satisifes requirements (1)–(4). Its similarity to the explicandum is clear, although Piscis is a narrower concept than Fish, owing to the exactness of the concept Piscis. It is fruitful, since it allows for more general and law-like statements about it than does the concept Fish. And finally it is simple, being defined only by the three properties 'lives in water', 'is a cold-blooded vertebrate', and 'has gills throughout life', each of which is easy to identify and confirm.

6.1.2 Classificatory, Comparative, and Quantitative Concepts

SUMMARY. We elaborate upon Carnap's discussion in [5, §I.4] of *classificatory*, *comparative*, and *quantitative* concepts.

— § —

Carnap defines the following types of concepts in [5, p. 8]:

> A *classificatory* concept (e.g., Warm) serves for classifying things into two kinds. A *comparative* concept is a relation based on a comparison, with the sense of 'more (in a cretain respect' (e.g., Warmer) or 'more or equal'. A *quantitative* concept serves to describe something with the help of numerical values (e.g., temperature).

Carnap discusses how a process of explication will often lead from a classificatory concept to a comparative concept, and from a comparative concept to a fully scientific quantitative concept. For example, we have the classificatory concept Warm, which classifies things into two classes: one class consisting of those items which are warm, and the other class those which are not warm. Next, we have the comparative concept Warmer, which is satisfied for a pair (x, y) whenever x is warmer than y. Then finally, we have a temperature $t(x)$ associated with each object x. One way to see the problem is that, for some universe of discourse U, a classificatory concept picks out a subset of U, a comparative concept endows U with (relational) structure, and a quantitative concept provides specific measurements to the elements of U.

Another way to look at the problem is to see each type of concept as a particular kind of mapping. A classificatory concept is a map $C : U \to \Omega$, where $\Omega = \{\text{True}, \text{False}\}$. Specifically, for an element $u \in U$, $C(u) = \text{True}$ when u satisfies the concept and $C(u) = \text{False}$ when u does not satisfy the concept. On the other hand, a comparative concept can be conceived as a map $C : U \to N$, where N is some ordered set. Since Carnap conceives of comparative concepts as constituting 'more (or equal to)' relations, then N is some totally ordered set. Thus, a comparative concept associates the elements of U with elements in an ordered set. Finally, a quantitative concept is a map $C : U \to T$ where T is, for instance, a field or vector space over e.g. the real numbers.

$$\begin{tikzcd} U \arrow[d, "Warm_U"] \\ \Omega \end{tikzcd}$$

(a)

$$\begin{tikzcd} U \arrow[r, "f"] \arrow[dr, dashed] & S \arrow[d, "Warm_S"] \\ & \Omega \end{tikzcd}$$

(b)

$$\begin{tikzcd} \mathbf{Limit}(U \xrightarrow{f} S) \arrow[d, "Warm_f"] \\ \Omega \end{tikzcd}$$

(c)

Figure 6.1

When the three kinds of concepts are seen in this perspective, we can see why the types of concepts are ordered in terms of their degree of explicitness: The structure of N is more fine-grained than the structure of Ω, and the structure of T is more fine-grained than the structure of N. Therefore comparative concepts are more scientific than classificatory concepts, and quantitative concepts are more scientific that comparative concepts. In the case of Ω, we have an unstructured set that simply has two truth values. In the case of N, we have a set given the structure of a total order, which is clearly more enriched than Ω. Finally, T not only contains the order information of N, since number systems are naturally ordered, but we also have algebraic operations defined on T, such as addition, subtraction, and maybe multiplication and division. Thus, we not only are able to say e.g. that x is less than y, as we would in N, but we are able to say *by how much* x is less than y. Thus, the passage from classificatory to comparative to quantitative concepts is the passage to increasingly fine-grained structures.

Thus, the concept Warm goes through a series of transformations: first, as a classificatory concept, it classifies those things which are Warm. As a comparative concept, Warmer provides an ordering to things based on their relative warmth. Finally, as a quantitative concept, Temperature provides a specific quantitative value to things, which is a measure of their actual temperature. But more specifically, once Temperature is introduced, then the meaning of the concept Warmer is meant in the sense of 'having a higher temperature'.

Although this seems clear enough, there is something lurking beneath this example which is significant. Notice that to have a concept Warm at all, there must be some sort of measurement applied to objects of U. For instance, if I say "x is Warm", then I am passing from an object x to a bodily sensation of warmth s. There are three ways to interpret this with regards to *what* specifically has the property Warm. The first is the traditional view, that x has the property Warm. Another view is that it is the sensation s itself, caused by coming into contact with x, that is Warm. Finally, there is the view that it is the very relation between x and s that is Warm. These three views furnish the three diagrams in Figure 6.1.

Perhaps these three views could be associated with three levels of consciousness. The

$$\begin{tikzcd}
U \arrow[r, "f"] \arrow[dr, "Warm^*_S"'] & S \arrow[d, "Warm_S"] \\
 & \Omega
\end{tikzcd}$$

(a) Classificatory concept (C_1).

$$\begin{tikzcd}
U \times U \arrow[r, "(f{,}f)"] \arrow[dr, "Warmer^*_S"'] & S \times S \arrow[d, "Warmer_S"] \\
 & \Omega
\end{tikzcd}$$

(b) Comparative concept (C_2).

$$\begin{tikzcd}
U \times U \arrow[r, "(g{,}g)"] \arrow[dr, "Warmer^*_T"'] & T \times T \arrow[d, "Warmer_T"] \\
 & \Omega
\end{tikzcd}$$

(c) Quantitative concept (C_3).

Figure 6.2

first is level conceives that objects themselves have properties. The second view is that it is not objects, but one's perceptions of objects that have properties. The third view is that the *relation* between objects and one's perceptions of them is what bears properties. While the third seems perhaps the most compelling philosophically, it will not be of much practical use in the rest of the chapter. However, I claim that the second view brings to light an important aspect of predication. Note that in Figure 6.1b we still recover an extension via $Warm_S \circ f : U \to \Omega$, so we are not merely defining properties on sensations. However, the fact that we have an intermediate S between U and Ω is crucial, as it incorporates the notion of *measurement* into the process of property-ascription. Thus, it is not enough to say simply that x has some property P, but we must ask *according to what measurement* does x have the property P. In the case above, an object $x \in U$ is $Warm$ under the condition that $Warm_S(f(x)) = \text{True}$. But this means (1) that the criterion for warmth is first defined on a measurement space S, which here is our bodily sensations, and (2) that our measurement device f yields a value in S that is $Warm_S$.

Seen in this light, we may then understand the classificatory (C_1), comparative (C_2), and quantitative (C_3) concepts of Warm as modeled by the diagrams in Figure 6.2.

It is clear why a comparative concept is more 'rich' than a classificatory concept, as it endows U and S with some order structure. The move from comparative to quantitative requires more explanation. Since we think of S as a set of bodily sensations, and T as an algebraic structure (with numbers as elements), then it seems that T is more structured than S, which could potentially explain why it is more 'scientific'. However, if we were to investigate S in depth, the problem would likely not be that it is poor in structure, but on the contrary that its structure is *too complex*. A mathematical space S (which would constitute a structure, in our theory) that sufficiently models sensation would likely be an immensely complex structure. Also, there would very likely need to be a unique S for each individual, since humans are quite different from one another. So really, the problem with S being used as a measurement space is that it is both far too complex and not well-understood by a scientific community. There using S as a measurement space is unfruitful when taking into account the fact that scientific consensus requires structures and laws that are easily identifiable by scientific communities.

Another problem that hinges on the fact that S is too complex of a measurement space is that, due to the complexity of S, the measuring apparatus f must also be convoluted, and likely not reproducible in any obvious way. So in reality, f probably changes depending on too many variables, such as e.g. the temperature of the room that the person was in prior to entering a new room. On the other hand, the measuring apparatus g, which could be e.g. a thermometer, works in a much simpler way, since the relation between the physical body and the numerical measurement of the thermometer is established by well-known causal laws.

So we have identified two reasons why the quantitative concept is more scientific than the classificatory and comparative concepts: (1) the measurement space T is better understood than S, and (2) the measurement device g functions according to well-known causal laws, whereas f does not. This to me seems to be the reason that C_3 is more scientific than C_2 and C_1. Thus, it doesn't seem like a crucial requirement that T be some numerical structure, nor that $Warmer_S \subset S \times S$ be some total order. Instead, it seems more generally that the following should hold:

1. T is better understood by a scientific community than is S.
2. The measuring apparatus g operates according to well known rules (such as causal laws), whereas the operation of f is less well-understood.[2]

The first point takes care of Carnap's criterion of exactness, whereas the second point takes care of his criterion of simplicity.

The use of a measurement space is also significant insofar as it allows us to define properties and relations 'in the abstract', without requiring extensions. For instance, $Warmer_T$ is a relation that applies to the numerical temperature values, not temperatures of objects themselves. This thus provides a criterion for what a pair (u, v) of objects or locations must satisfy in order for u to be warmer than v. Specifically, u is warmer than v if $g(u) > g(v)$. But we can change the domain U to any arbitrary domain U', and this choice of domain does not change the meaning of Warmer, insofar as no matter the domain, it must be measured by some g that gives values in T, and the relation $Warmer_T \subset T \times T$ is fixed. So by fixing a measurement space T, we are able to preserve meaning even while extensions

[2]In Section 6.2.4.2, we will return to this problem by defining the *descriptive complexity* of a function.

change. In Section 6.2.4.1, we will define such predicates that apply to these measurement spaces (which we later call *evaluation spaces* for the sake of generality) as *abstract predicates.*

6.1.3 Our Project: Enriching Traditional Semantical Approaches

SUMMARY. We provide an overview of the traditional (Tarskian) approach to semantics, along with its generalizations provided by categorical semantics. We then specify three objectives of this chapter: (1) how to move from 'global' to 'local' objects of universes of discourse, (2) how to construct generalized truth values objects, and (3) how to conceive of the semantics of predicates independently of their extensions.

— § —

In traditional (Tarskian) semantics, properties are defined with respect to a set U of individuals. The set U is called a *universe of discourse*, a *universe of objects*, or more simply a *universe.* For some property P, the meaning of P is given by its extension over U, and thus properties are identified with subsets of U. The same is true for an n-ary relation R, which gives a subobject of U^n, namely those n-tuples of individuals that satisfy the relation. In a classical logic, a property P is interpreted as a mapping $[\![P]\!] : U \to 2$, where 2 contains the truth-values 0 and 1, denoting False and True, respectively. Thus, the statement $P(x)$ is true iff $[\![P]\!](i_x) = 1$, where $i_x \in U$ is the individual denoted by the constant symbol x in the language.

Truth-evaluation can be generalized to the situation where truth values are not merely 0 and 1. For instance, in the logic of a sheaf topos over a topological space X, the truth values consist of the open sets of X. This endows the logic with a notion of 'local truth': propositions in this logic are not absolutely true or absolutely false, but true in some places and not true in others. For instance, if X is a topology on $\mathbb{R}$, where $\mathbb{R}$ represents time, then a proposition such as 'John Smith is a US senator' is true for any open set V in X during which John Smith is a senator for the entirety of V. In a categorical context, the 'collection' of truth values of a certain logic is called a subobject classifier, and is denoted by Ω.

The situation can also be generalized so that the universe of discourse U is not simply a set, but some other object, such as an algebraic structure. Also, there may be multiple universes, not just one U. In model theory, the syntactic correlates of the semantical universes are called *sorts*, and can be built up into compound sorts, called *types.* The idea is that a sort or type is a universe *in the language*, whereas its semantic interpretation is some object *outside the language.* The requirements for a category $\mathscr{C}$ to be able to provide the semantics for some (first-order) language L are that $\mathscr{C}$ has (1) arbitrary finite products, and (2) a subobject classifier. The existence of products is what enables one to define n-place predicates, whereas the existence of a subobject classifier is what enables one to classify the individuals that satisfy the predicates.

Thus we provide the definition from [11, p. 808] of a *signature*:

Definition 6.1 (Signature)**.** A *signature* Σ is a triple $\Sigma = (\Sigma\text{-Sort}, \Sigma\text{-Fun}, \Sigma\text{-Pred})$, where:

1. Σ-Sort is a set of sorts.

2. Σ-Fun is a set of function symbols, i.e., maps $f : A \to B$ between sorts.

3. Σ-Pred is a set of predicate symbols.

Having defined what constitutes a signature Σ, we can define what constitutes an interpretation of Σ, which is called a Σ-structure[3]:

Definition 6.2 (Σ-structure)**.** Let Σ be a signature, and $\mathscr{C}$ a category with finite products and a subobject classifier Ω. A Σ-structure is given by a map M, where:

1. For each sort $A \in \Sigma$-Sort, MA is an object in $\mathscr{C}$. For a string $A_1, \ldots, A_n$ of sorts, then $M(A_1, \ldots, A_n) = MA_1 \times \cdots \times MA_n$. (For the empty string $[]$, then $M([]) = 1$, the terminal object in $\mathscr{C}$.)

2. For each function symbol $f : A_1 \cdots A_n \to B \in \Sigma$-Fun, $Mf : A_1 \times \cdots \times A_n \to B$ is a morphism in $\mathscr{C}$.

3. For each predicate symbol $P \rightarrowtail A_1 \cdots A_n \in \Sigma$-Pred, MP is a subobject $MP \rightarrowtail M(A_1, \ldots, A_n)$ in $\mathscr{C}$. (A predicate P likewise corresponds to a morphism $[\![P]\!] : M(A_1, \ldots, A_n) \to \Omega$ in $\mathscr{C}$.)

Thus we have two generalizations of classical semantics, which is achieved by moving from the category **Set** of sets to some other category $\mathscr{C}$: (1) set-theoretic domains of discourse U can be generalized to objects other than sets, and (2) the Boolean subobject classifier 2 can be generalized to a more complex subobject classifier Ω. In Section 6.2.3, we will deal with a format that provides great freedom in constructing objects that behave like subobject classifiers. This will, in turn, furnish great freedom in constructing languages.

There is still something that neither of these generalizations can account for, however, which is the fact that individuals $u \in U$ may themselves have structure, and therefore it seems too strong to say that some u satisfies, *in its entirety*, some property P. For instance, if U consists of pieces of music, and I make the assertion that "u is Loud", then this does not necessarily mean that every moment in u is loud. This seems like a problem of 'local truth', and therefore we might be led to assume that a sheaf topos could model the assertion "u is Loud". However, this is not correct, since we want to evaluate the proposition "u is Loud" *relative to u itself*, not relative to some global ambient space in which u exists. Specifically, we want to interpret the predicate Loud so that it applies to the *parts* of each piece $x \in U$ that are Loud. So, this is a different type of relativity than

[3]This notion of structure is distinct from the notion of structure that we defined in Section .

that encountered in sheaf toposes, where a proposition is true *relative to some location in the global space X*. In the situation that we are attempting to outline, a proposition $P(u)$ is true *in the locations of u itself* where the proposition holds. We will deal with this situation in Section 6.2.2.

A final objective of this chapter will be to conceive of a way to derive the semantics of predicates that are, in some sense, independent of extensions. This relates to the discussion in Section 6.1.2, where the measurement space T for temperature allows us to define the property $Warmer_T$, which is independent of the extension of the property Warm over a universe of discourse. Defining these *abstract predicates* will enable us to reason about *potential* objects, rather than only *actual* objects. We deal with this issue in Section 6.2.4.

6.2 Tools for Language-Construction

Summary. We provide tools that enable construction of precise predicates in any context. This involves defining (1) subobject predication, (2) truth values objects, and (3) abstract predicates which provide a means for dealing with semantics independently of extensions. We first provide some preliminary constructions that will be used throughout the chapter.

— § —

6.2.1 Preliminary Constructions

Summary. We provide two kinds of constructions, each of which enables us to more easily speak of collections of structures.

— § —

In the following, when we speak of structures, assume that this includes compositions via the canonical encoding of compositions as structures (see Section 3). To construct languages for reasoning about structures, we want to have sets of structures for universes of discourse. There are two methods for acquiring sets of structures: (1) we can translate sets of structures into structures, or (2) we can translate structures into sets. We can achieve (1) via the following

Convention 6.1. Let $U = \{S_i\}_{i \in I}$ be a set of structures. We can encode U as a structure $Str(U)$ via the following sequence of procedures:

1. For each $S_i \in U$, derive the structure

$$Pow(S_i) \xrightarrow[\mathrm{Id}]{} \mathbf{Power}(S_i).$$

2. For each $Pow(S_i)$, define the morphism $\top_i : 1 \to Pow(S_i)$ that picks out the top element S_i of $Pow(S_i)$. Derive therefore the limit structure

$$S_iElem \underset{\mathrm{Id}}{\longrightarrow} \mathbf{Limit}(1 \xrightarrow{\top_i} Pow(S_i)).$$

   S_iElem corresponds to the singleton set $\{S_i\}$.

3. Next, define

$$Str(U) \underset{\mathrm{Id}}{\longrightarrow} \mathbf{Colimit}(S_1Elem, \ldots, S_nElem).$$

   $Str(U)$ is very much like the actual set U, since $Str(U)$ corresponds to taking the disjoint union of the structures S_iElem, which are themselves like singleton sets. Thus, $Str(U)$ corresponds to taking the disjoint union of the singleton sets of structures $\{S_1\}, \ldots, \{S_n\}$, which is of course equal to U.

Although this method works for translating structures into sets, it is rather bloated. So instead, we will take the opposite approach and translate structures into sets.

Our first step in achieving this translation is to define what we will call a *unit of structure*. Units of structure can be thought of as the elementary building blocks of structures, as we will see.

Recall that for a structure S and set A in **Str**, the set $A@S$ consists of the A-addressed points of S. Acquiring the elements of $A@S$ depends on the type of S, as follows:

1. If $\mathrm{Type}(S) = \mathbf{Simple}$, with coordinator $@X$, then $A@S \subseteq \mathrm{Hom}_{\mathbf{Rel}}(A, X)$.

2. If $\mathrm{Type}(S) = \mathbf{Limit}$, with coordinator $\mathcal{D}$ consisting of coordinators $C_1, \ldots, C_n$, then $A@S \subseteq \prod_{i=1}^n A@C_i$. (The subset is determined by the morphisms in $\mathcal{D}$. If $\mathcal{D}$ is a discrete diagram, then $A@S = \prod_{i=1}^n A@C_i$.)

3. If $\mathrm{Type}(S) = \mathbf{Colimit}$, with coordinator $\mathcal{D}$ consisting of coordinators $C_1, \ldots, C_n$, then $A@S = \left(\coprod_{i=1}^n A@C_i\right)/\sim$. (The equivalence relation $\sim$ is determined by the morphisms in $\mathcal{D}$.)

4. If $\mathrm{Type}(S) = \mathbf{Power}$, with coordinator C, then $A@S$ corresponds to $A@\Omega^C$. The set $A@\Omega^C$ is in bijection with the set $Sub(@A \times C)$ of subfunctors of $@A \times C$. Since $@A \times C$ is a product, then each subfunctor $\widehat{S} \subseteq @A \times C$ is likewise a product, and thus $A@\widehat{S} = A@A \times A@C$, as explained for when $\mathrm{Type}(S) = \mathbf{Limit}$.

5. If $\mathrm{Type}(S) = \mathbf{Sub}$, with coordinator C, then $A@S \subseteq A@C$.

6. If $\mathrm{Type}(S) = \mathbf{Hom}$, with coordinators C_1, C_2, then note that a morphism $f : C_1 \to C_2$ corresponds to a subobject of the product $C_1 \times C_2$. Thus, $C_2^{C_1}$ corresponds to a power structure

$$Funcs(C_1, C_2) \underset{Fu \rightarrowtail C_1 \times C_2}{\longrightarrow} \mathbf{Power}(C_1 \times C_2),$$

where Fu consists of all subobjects of $C_1 \times C_2$ that correspond to morphisms $f : C_1 \to C_2$. Thus $A@S$ corresponds to $A@Funcs(C_1, C_2)$. We have already confirmed how to take care of power structures, and $Funcs(C_1, C_2)$ is a power structure. Therefore follow the instructions for when $\text{Type}(S) = \textbf{Power}$.

Now we can define a *unit of structure*, as so:

Definition 6.3 (Unit of Structure)**.** Let S be any structure in **Str** (or, equivalently, any functor in $\mathbf{Rel}^{@}$) and A any set in **Rel**. A *unit of structure* is an element of $A@S$. Thus, varying over all sets A in **Rel** and all structures S in **Str**, we get the collection of units of structure $\mathcal{U}_0$ as

$$\mathcal{U}_0 = \bigcup_{A \in \mathbf{Rel}_0, S \in \mathbf{Str}_0} A@S. \tag{6.1}$$

From $\mathcal{U}_0$, we can easily recover structures. First of all, define the object map

$$H : \mathbf{Str}_0 \to \mathbf{Set}_0 : S \mapsto \bigcup_{A \in \mathbf{Rel}_0} A@S, \tag{6.2}$$

which, for a structure S, gives the set of S's units of structure. (H will be used throughout this chapter.) However, since S also has a name, we cannot simply identify it with $H(S)$. Thus, let $\mathfrak{N}$ be the namespace from which the names of structures are taken. We have the product $\mathfrak{N} \times \mathcal{P}(\mathcal{U}_0)$, consisting of pairs (N, U) where N is a name and U a subset of $\mathcal{U}_0$. We have a subset $\mathcal{U}_1 \subset \mathfrak{N} \times \mathcal{P}(\mathcal{U}_0)$ that corresponds to structures, where the first coordinate is the name of a structure S, and the second coordinate is the set $H(S)$. Thus structures $\mathbf{Str}_0$ are in bijective correspondence with sets in $\mathcal{U}_1$.

Finally, we get a set of sets of structures via $\mathcal{P}(\mathcal{U}_1) = \mathcal{U}_2$. So we have three levels:

1. $\mathcal{U}_0$, consisting of units of structure.

2. $\mathcal{U}_1$, consisting of structures.

3. $\mathcal{U}_2$, consisting of sets of structures.

In many sections in this chapter, it will be convenient to assume that structures are elements of $\mathcal{U}_1$, and sets of structures are elements of $\mathcal{U}_2$. We will denote our usual structures as **Str**-structures, and the structures in $\mathcal{U}_1$ as **Set**-structures. However, if the context is clear for whether we are dealing with **Str**-structures or **Set**-structures, then we will drop the prefix.

We can also translate **Str**-structure morphisms into **Set**-structure morphisms. This is an easy translation, and is left as an exercise.

6.2.2 Local Predication

Summary. In classical semantical situations, the objects of universes of discourse have no deeper structure. We expand upon that view by defining *attributes*. An attribute is either 'analytic' or 'synthetic': an analytic attribute is a rule for extracting 'local objects' from 'global objects'; a synthetic attribute is a rule for endowing objects with additional information. The definitions of analytic and synthetic attributes paves the way for what will be defined as *descriptive enrichments*. All of these steps enable us to predicate properties of objects that apply to *aspects* of those objects, instead of the naive view that a property applies to the entirety of an object.

— § —

6.2.2.1 Local Attributes of Structures

Summary. We discuss analytic and synthetic attributes. We will see that a synthetic attribute is often dependent on an analytic attribute.

— § —

In the following, we define and discuss *attributes* of structures. By an *attribute* of a structure is meant the intuitive idea of an aspect or trait, such as 'color', 'shape', and so on. There are two types of attributes that we wish to discuss, namely *analytic* attributes and *synthetic* attributes. The terms 'analytic' and 'synthetic' are here meant in the philosophical tradition of Kant, where an analytic truth is true by definition and a synthetic truth is true by some external criterion. The idea of an analytic attribute of a structure is that the attribute is in some way encoded in the structure's definition. On the other hand, a synthetic attribute of a structure is an attribute that it has externally to its definition.

(I) *Analytic Attributes.* Let us first discuss analytic attributes. Let S be a structure. Recall from [chapter and section] that its ramification tree is the graphical representation of its hierarchical construction. For instance, define as in [15] the structure

$$Note \underset{\mathrm{Id}}{\longrightarrow} \mathbf{Limit}(Onset, Pitch, Loudness, Duration), \tag{6.3}$$

with lower level structures

$$Onset \underset{Fu_1 \rightarrowtail @\mathbb{R}}{\longrightarrow} \mathbf{Simple}(\mathbb{R}), \tag{6.4}$$

$$Pitch \underset{Fu_2 \rightarrowtail @\mathbb{Z}}{\longrightarrow} \mathbf{Simple}(\mathbb{Z}), \tag{6.5}$$

$$Loudness \underset{Fu_3 \rightarrowtail @\mathbb{R}}{\longrightarrow} \mathbf{Simple}(\mathbb{R}), \tag{6.6}$$

$$Duration \underset{Fu_4 \rightarrowtail @\mathbb{R}}{\longrightarrow} \mathbf{Simple}(\mathbb{R}), \tag{6.7}$$

$$\tag{6.8}$$

each of which defines a module.

Given the structure $Note$, we may wish to inspect its onset information. In this case, Onset would constitute an analytic attribute of $Note$, since $Onset$ is constitutive of the definition of $Note$. Since $Note$ is a simple product of structures, we can recover $Onset$ via projection $\mathrm{pr}_1(Note)$. Notice, however, that if there were morphisms between the coordinators, then projection onto $Onset$ would not necessarily return all of $Onset$, but a subobject $K \subset Onset$. Nonetheless, the $Onset$ 'material' that is present in $Note$ is still recoverable, so there is nothing lost when performing the projection.

However, for colimit and power structures, there is no such projection, so we need to discuss the general situation for recovering the lower level information of a structure. The first method that we will discuss is a bit messy. However, it allows us to say that an attribute α of a structure S is, up to isomorphism, one of its subobjects $S_\alpha \subset S$.

Lemma 6.1. *Let S and T denote structures, and define*

$$Lim \underset{\mathrm{Id}}{\longrightarrow} \mathbf{Limit}(S, T).$$

We can recover $\mathrm{pr}_1(Lim)$ (resp. $\mathrm{pr}_2(Lim)$) up to isomorphism as a subobject $\widehat{Lim} \subset Lim$.

Proof. Recall the mapping H from Equation 6.2. Note that Lim corresponds to a set of pairs, via $H(Lim) = \{A@S \times A@T : A \in \mathbf{Rel}_0\}$. Therefore $\mathbf{Power}(Lim)$ corresponds to the subsets P of $H(Lim)$ such that $\mathrm{pr}_1(P)$ is a substructure of S and $\mathrm{pr}_2(P)$ is a substructure of T. Thus, simply take the subset P_{pt} where $\mathrm{pr}_1(P_{\mathrm{pt}}) = S$ and $\mathrm{pr}_2(P_{\mathrm{pt}})$ is the substructure of T given by taking only a single functional 1-addressed point $\mathrm{pt} \in 1@T$ (the other morphisms being inferred), thus reducing the structure of T to a single one of its elements. Since $\mathrm{pt} \cong 1$, the effect is that of returning $S \times 1$. Thus we have $S \times 1 \cong S$. $\square$

The situation is easily generalized to limits of arbitrary diagrams of structures. For structures of type **Colimit**, **Power**, and **Sub**, the situation is easy to verify. For colimit structures, the subobjects corresponding to the coordinators are the canonical injections; for power structures, the coordinator structure is recovered by taking the top element $\top$ of the power structure; for sub structures, the situation is obvious. For structures of type **Hom**, we can recover the coordinators C_1, C_2 via similar procedures as those discussed in the sixth item of the second list in Section 6.2.1.

However, instead of proceeding to recover the lower level structures in this way, we provide an alternative method. Specifically, we have the following recursive methods.

1. **Limit**. For a limit structure S with coordinators $\{C_i\}_{i \in I}$, to get the coordinator C_k, perform the kth projection $\mathrm{pr}_k(S)$. Note that $\mathrm{pr}_k(S)$ does not necessarily equal C_k. This is because of the morphisms that may be present in the coordinator $\mathcal{D}$ of S. For instance, if

$$S \underset{\mathrm{Id}}{\longrightarrow} \mathbf{Limit}(1 \xrightarrow{p} C),$$

where $p(1)$ picks out a single point from C, then S consists only of the pair $(1, p(1))$. Thus, $\mathrm{pr}_2(S) = p(1)$, which is a single element of C.

2. **Colimit**. For a colimit structure S with coordinators $\{C_i\}_{i \in I}$, we automatically get the coordinator C_k, since the colimit always preserves all information. This is because, even if we have morphisms between the coordinators, then this simply makes equivalence classes of elements in the disjoint union of the coordinators, and therefore the elements persist no matter what. Note, however, that if we have arrived at S after projecting out of some limit structure T of which S is a coordinator, then the totality of S may not be present. This is a result of what was discussed in the previous point, regarding morphisms in the coordinator of a limit structure. For instance, if we have

$$T \underset{\mathrm{Id}}{\longrightarrow} \mathbf{Limit}(1 \xrightarrow{p} S),$$

then $\mathrm{pr}_2(T) = p(1) \subset S$, which is, as above, a single element (potentially an equivalence class) of S. Nonetheless, in this case we can still easily recover what is left over via a canonical injection. For instance, to recover the component of the kth coordinator C_k of S, then we get the morphism p_k that makes the triangle

$$\begin{tikzcd} 1 \arrow[r, "p_k"] \arrow[dr, "p"'] & C_k \arrow[d, tail, "i_k"] \\ & S \end{tikzcd}$$

commute.

3. **Power**. Again, what we can recover from a power structure depends on whether there are higher level limit structures out of which we have projected. For instance, suppose we have a structure

$$T \underset{\mathrm{Id}}{\longrightarrow} \mathbf{Limit}(X \xrightarrow{f} S),$$

where

$$S \underset{\mathrm{Id}}{\longrightarrow} \mathbf{Power}(A).$$

To recover S from T, we first perform $\mathrm{pr}_2(T)$. However, if $A \notin f(X)$, then we cannot recover A from the projection. But we would still like to recover as much of A as possible from $\mathrm{pr}_2(T)$. To do this, we then take the join of all the elements in the subobject hierarchy of $\mathrm{pr}_2(T)$, which is given by

$$\bigcup_{\widehat{A} \in Sub(\mathrm{pr}_2(T))} \widehat{A}.$$

4. **Sub**. This situation is obvious.

5. **Hom**. As demonstrated previously, we can recover the coordinators of a hom structure
$$S \xrightarrow[\mathrm{Id}]{} \mathbf{Hom}(C_1, C_2)$$
via translation of S into a power structure
$$S' \xrightarrow[Fu \rightarrowtail C_1 \times C_2]{} \mathbf{Power}(C_1 \times C_2).$$
Thus follow the instructions for **Power** above.

With these five procedures outline, we are given instructions regarding how to 'excavate' an analytic attribute α from a structure S: we simply iterate procedures 1–5 above as we are descending through the ramification tree of S.

Note, however, that an ambiguity may arise. Suppose we want to recover some α-attribute from S. Then this attribute should correspond to a (subobject of a) structure S_α in S's ramification tree. However, what if there are multiple copies of S_α? For instance, suppose
$$S \xrightarrow[\mathrm{Id}]{} \mathbf{Limit}(F_\alpha, F_\alpha).$$
Then if we want to recover an α attribute of S, we are left with a choice between the first and second projection. Therefore, as a convention, when seeking to excavate an α attribute, we should provide a means of identification of the nodes in the ramification tree. This is simple, as it simply consists in assigning an integer sequence to each node, which can be done canonically in the following way, depending on the type of S. First of all, since S is at the top of the ramification tree, it receives the integer 1. Then the following recursive procedures provide the method for assigning integer sequences to the nodes of S's ramification tree:

- If S is a limit, colimit, or hom structure, its coordinators $C_1, \ldots, C_n$ are ordered in sequence, as given by their indices. Then $\mathrm{pr}_i(S)$ is given the integer sequence $1.i$.
- If S is a power or sub structure, then moving down one level to its coordinator A (or a subobject of A) gives 1.1.

Thus, for an attribute α, we can define a map $f_\alpha : U \to U_\alpha$ from a set U of structures to a set U_α of structures derived from U according to the 'excavation' procedures outlined above. If a structure $S \in U$ has no α-attribute, then $f_\alpha(S)$ simply returns the empty structure $\varnothing$.

(II) *Synthetic Attributes.* A synthetic attribute of a structure S is an attribute that is not contained in S's ramification tree. To give an informal example, suppose we have some encoding of a physical object x, and we wish to gauge its temperature. In this case, Temperature is likely not an aspect involved in the construction of x, but nonetheless one should be able to identify a temperature of x, since it's an encoding of a physical object and physical objects have a temperature. (We will see in Section 6.2.4 that synthetic attributes relate to the notion of an *evaluation space.*)

To conceive of a synthetic attribute σ of a **Set**-structure S, we require a **Set**-structure M and a map that sends S (or an analytic attribute S_α of S) to an element $m \in M$. Thus, for a set U of structures, we can acquire their synthetic attribute σ as a map

$$\sigma : U \to M. \tag{6.9}$$

In many situations, however, it is more precise to define a synthetic attribute so that it applies not to the totality of S, but to a *part* of S, where a part is given by an analytic attribute S_α. Therefore we may wish to define some map $\sigma_\alpha : \alpha(U) \to M$, which thus gives σ via the commutativity of

$$\begin{tikzcd}
U \arrow[r, "\alpha"] \arrow[dr, "\sigma"'] & \alpha(U) \arrow[d, "\sigma_\alpha"] \\
 & M
\end{tikzcd}$$

Now that we've defined analytic and synthetic attributes, when defining predicates on structures we have the extra explicatory power of specifying which *aspect* (given by an attribute) of the structure satisfies the predicate. This enables us to predicate properties/relations of an object *relative to locations of the object itself.*

6.2.2.2 Descriptive Enrichments of Structures

Summary. We present the notion of a *descriptive enrichment.* These are meant to provide more fleshed out descriptions of structures and compositions.

— § —

A *descriptive enrichment* is a means of describing an object in terms of its attributes. Recall that analytic and synthetic attributes are special kinds of maps. An analytic attribute is a map $\alpha : U \to U_\alpha$ from a set U of structures to a set U_α of structures that are in some sense 'substructures' of those structures in U. On the other hand, a synthetic attribute is a map $\sigma : U \to M$, possibly with $\sigma = \sigma_\beta \circ \beta$ for $\beta : U \to U_\beta$ an analytic attribute and $\sigma_\beta : U_\beta \to M$ a synthetic attribute that applies to the β-parts of each $S \in U$. Thus, for a set U of structures, a set $\mathbb{A}$ of analytic attributes, and a set $\mathbb{S}$ of synthetic attributes, we can use all of these attributes $\Xi = \mathbb{A} \sqcup \mathbb{S}$ to provide more thorough descriptions of the structures in U. Also, the fact that we can use the same set Ξ of attributes and apply all the maps $\xi \in \Xi$ to every $S \in U$ provides a uniformity with regards to the descriptions of the structures in U.

Note that for a set Ξ of attributes defined on U, and a structure $S \in U$, we have the set

$$\Xi(S) = \{\xi(S) : \xi \in \Xi\}. \tag{6.10}$$

We would like to be able to classify the subsets of Ξ, and then apply such classes of morphisms to each $S \in U$. Moving 'upward', we would like to provide classifications to the subsets of Ξ, and so on to arbitrary level n. Thus we define the following:

Definition 6.4 (Descriptive Enrichment Scheme)**.** A *descriptive enrichment scheme* is a quadruple $\mathscr{D} = \left(\Xi, \mathfrak{N}, \overrightarrow{C}, \overrightarrow{N}\right)$, where:

1. Ξ is a set of attributes defined over some domain U of structures.

2. $\mathfrak{N}$ is the namespace, used to name individuals and classes.

3. $\overrightarrow{C} = \left(C_1, \ldots, C_n\right)$, where $C_1 = \Xi$, and each C_{i+1} is a subset $C_{i+1} \subseteq \mathcal{P}(C_i)$ such that C_{i+1} covers C_i. (Note that each C_i can be thought of as an ith order property, and therefore the first-order domain of discourse is $C_1 = \Xi$.)

4. $\overrightarrow{N} = \left(N_1 : C_1 \to \mathfrak{N}, \ldots, N_n : C_n \to \mathfrak{N}\right)$, and thus each N_i provides names to the classes in C_i.

Essentially, $\mathscr{D}$ provides a hierarchic classification scheme whose lowest level of individuals is the set Ξ of attributes. This is fruitful in cases where we would like to classify attributes hierarchically. For instance, suppose that Ξ consists of attributes that are meant to provide information regarding color, shape, spatial coordinates, and temporal coordinates. Then we may wish to classify color and shape together – since they both apply to physical appearance –, and classify the spatial and temporal coordinates together – since they provide spatio-temporal information. Thus we'd get the set

$$C_2 = \{\{\text{color, shape}\}, \{\text{spatial coordinates, temporal coordinates}\}\} \subseteq \mathcal{P}(\Xi),$$

with e.g.

$$N_2(C_2) = \{\text{Appearance, Space-Time}\}.$$

We could probably stop here, but theoretically we could proceed to the nth level for arbitrary $n \in \mathbb{N}$.

Once we have a descriptive enrichment scheme $\mathscr{D}$, we can use it to instantiate how the classification scheme operates on some $X \in U$. First, recall that every structure F has a name $Name(F)$. Now we can define the following:

Definition 6.5 (Descriptive Enrichment)**.** Let $\mathfrak{D}$ be a descriptive enrichment scheme, and $S \in U$ a structure. A *descriptive enrichment* of S is denoted by $\mathfrak{D}(S)$, and the idea is that $\mathfrak{D}(S)$ instantiates $\mathfrak{D}$ by applying it to S. We set $\mathfrak{D}(S) = (\Xi, \mathfrak{N}, \overrightarrow{C}(S), \overrightarrow{N}(S))$, where:

1. Ξ is as in Definition 6.4.
2. $\mathfrak{N}$ is as in Definition 6.4.
3. Recall that $C_1 = \Xi$. Then define
$$\overrightarrow{C}(S) = \big(C_1(S), \ldots, C_n(S)\big),$$
where $C_1(S)$ is as defined in Equation 6.10, and each $C_{i+1}(S)$ is a subset $C_{i+1}(S) \subseteq \mathcal{P}(C_i(S))$.
4. Define the binary operation $\star : \mathfrak{N} \times \mathfrak{N} \to \mathfrak{N}$, where for strings $s, t \in \mathfrak{N}$, $s \star t$ is the concatenation of s and t. Then define
$$\overrightarrow{N}(S) = \big(N_1(S) : C_1(S) \to \mathfrak{N}, \ldots, N_n(S) : C_n(S) \to \mathfrak{N}\big),$$
where for each $K \in C_i$ we get the corresponding $K(S) \in C_i(S)$, so that
$$N_i(S)\big(K(S)\big) = \text{Name}(S) \star N_i(K).$$
Though this looks cumbersome, all that it does is the same thing as in Definition 6.4, but ensures that the name of S is attached to the names of each class in C_i. For instance, let $\text{Name}(S) = \mathit{thisStructure}$ and suppose a map $\xi \in C_1$ provides color information, and we have $N_1(\xi) = \text{Color}$. Then $N_1(S)\big(\xi(S)\big) = \mathit{thisStructure}\text{Color}$, which simply establishes that $\xi(S)$ is the Color attribute of *thisStructure* specifically.

We can also take a different approach to defining descriptive enrichments, by defining an nth-order signature Δ as follows:

1. We start with $C_1 = \Xi$ as defined above as our set of attribute maps, which will be brought into Δ as a sort $\mathbf{C}_1$. The names of the terms of $\mathbf{C}_1$ are the names of constant symbols in Δ, i.e., names of attribute maps (this corresponds to the naming map $N_1 : C_1 \to \mathfrak{N}$ in Definition 6.4).
2. The classification C_2 of C_1 in $\mathfrak{D}$ is brought into Δ by giving a set of property symbols
$$\{P_1^{\mathbf{C}_1}, \ldots P_{k_1}^{\mathbf{C}_1}\} = \Pi_1.$$
Their names, which were provided by $N_2(C_2)$ in Definition 6.4, are provided by the names of the property symbols in Δ. Thus, for each property symbol $P_i^{\mathbf{C}_1} \in \Pi_1$, we have the semantics of $P_i^{\mathbf{C}_1}$ as a subobject $[\![P_i^{\mathbf{C}_1}]\!] \rightarrowtail \mathbf{C}_1$.

3. The classification of Π_1 is given by a set of second-order property symbols
$$\{P_1^{\Pi_1}, \ldots, P_{k_2}^{\Pi_1}\} = \Pi_2,$$
their names again being provided by Δ. The semantics of each $P_i^{\Pi_1}$ is likewise given as a subobject $[\![P_i^{\Pi_1}]\!] \rightarrowtail \mathbf{C}_2$, where $\mathbf{C}_2 \rightarrowtail \Omega^{\mathbf{C}_1}$, and therefore $[\![P_i^{\Pi_1}]\!] \rightarrowtail \Omega^{\mathbf{C}_1}$

4. Proceed similarly for the classification of Π_2, and so on for Π_n.

Defining Δ sheds some light on the contrast between a descriptive enrichment scheme $\mathscr{D}$ and one of its instantiations $\mathscr{D}(S)$. Essentially, we can use the same signature Δ for dealing with $\mathscr{D}$ and $\mathscr{D}(S)$. The difference between $\mathscr{D}$ and $\mathscr{D}(S)$ corresponds to a difference in the *interpretation* of Δ. $\mathscr{D}$ will correspond to an interpretation M that sends $\mathbf{C}_1$ to the collection Ξ of attribute-maps, whereas $\mathscr{D}(S)$ will correspond to an interpretation $M(S)$ that sends $\mathbf{C}_1$ to the collection $\Xi(S)$ as defined by Equation 6.10. The contrast between M and $M(S)$ with regards to their interpretations of sorts Π_i consisting of predicate symbols naturally follows. (The only difference between the signature Δ for $\mathscr{D}$ and $\Delta(S)$ for $\mathscr{D}(S)$ would be in the names of the constant symbols and predicate symbols... but this is not an essential difference.)

The 'image' of such a signature Δ is given by Figure 6.3.

6.2.3 Truth Values

Summary. We provide the tools for creating truth values objects in **Str**. We follow the same procedures as Mazzola in [16].

— § —

In [16], Mazzola discusses the situation of creating languages for defining musicological predicates. In that text, it is shown how to create truth values objects with functors in the category $\mathbf{Mod}^@$ of set-valued presheaves over **Mod**, where the latter is the category of modules with affine homomorphisms. Mazzola calls these objects *forms*, and they are analogous to our *structures*.

In $\mathbf{Mod}^@$, there is the given subobject classifier Ω. However, one may wish to construct a truth values object with a richer collection of truth values. To construct such an object, we first pick a module I, and define the form
$$Val(I) \underset{\mathrm{Id}}{\longrightarrow} \mathbf{Simple}(I). \tag{6.11}$$

Then, to get our truth values, we construct
$$TRUTH(I) \underset{\mathrm{Id}}{\longrightarrow} \mathbf{Power}(Val(I)). \tag{6.12}$$

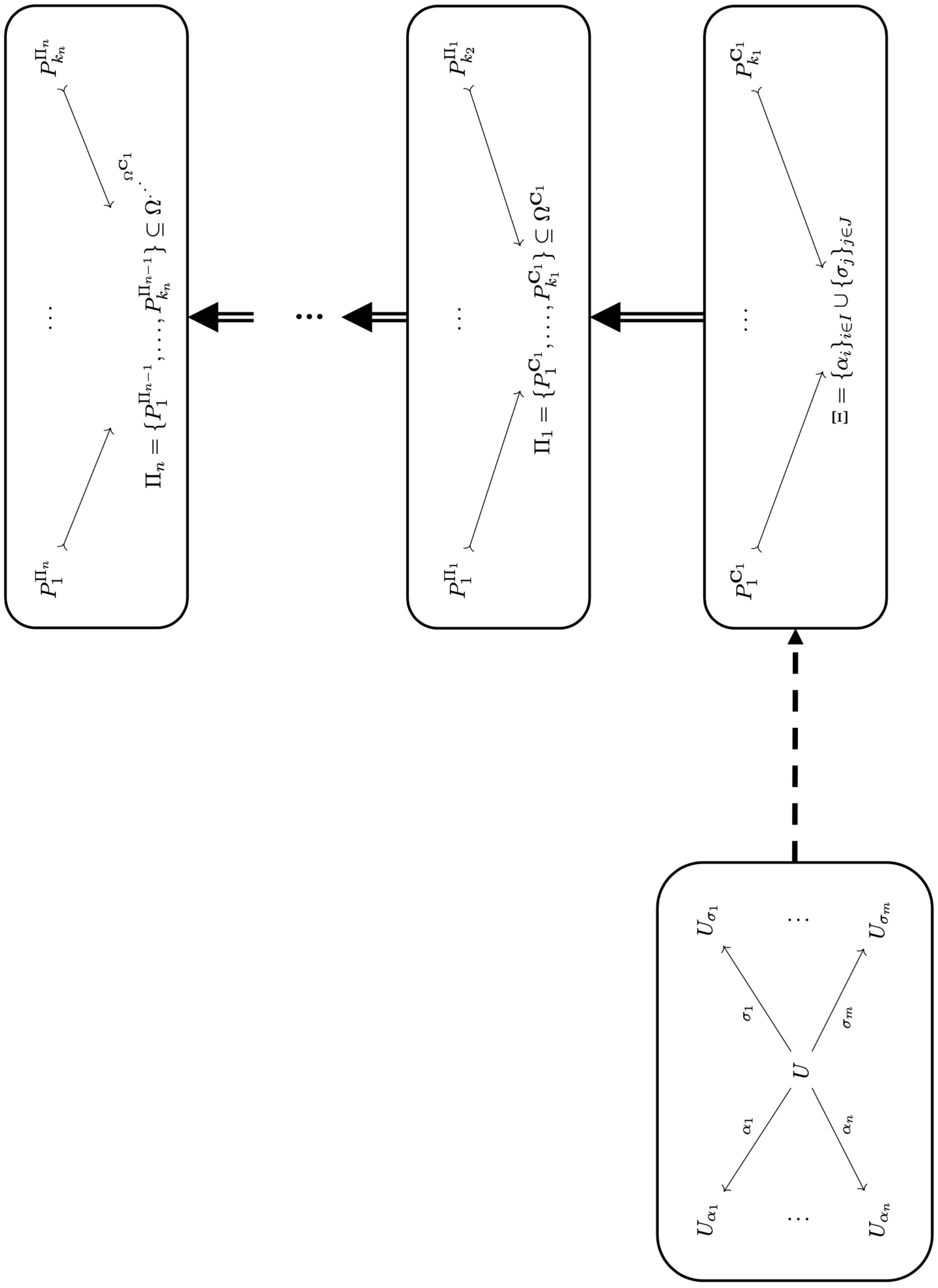

U
U_{α_1}
U_{α_n}
U_{σ_1}
U_{σ_m}
α_1
α_n
σ_1
σ_m
$P_1^{\mathbf{C}_1}$
$P_{k_1}^{\mathbf{C}_1}$
$\Xi = \{\alpha_i\}_{i \in I} \cup \{\sigma_j\}_{j \in J}$
$P_1^{\Pi_1}$
$P_{k_2}^{\Pi_1}$
$\Pi_1 = \{P_1^{\mathbf{C}_1}, \ldots, P_{k_1}^{\mathbf{C}_1}\} \subseteq \Omega^{\mathbf{C}_1}$
$P_1^{\Pi_n}$
$P_{k_n}^{\Pi_n}$
$\Pi_n = \{P_1^{\Pi_{n-1}}, \ldots, P_{k_n}^{\Pi_{n-1}}\} \subseteq \Omega^{\cdot^{\cdot^{\Omega^{\mathbf{C}_1}}}}$

Figure 6.3: A visual representation of the signature Δ. After extracting the attribute maps with domain U, we ascend through the classification hierarchy.

The alternate notation for $\mathbf{Power}(Val(I))$ is Ω^I, which expresses that $TRUTH(I)$ is the collection of subobjects of $Val(I)$, and these subobjects are to be used as truth values. As an example, Mazzola takes the real circle group $S_1 = \mathbb{R}/\mathbb{Z}$, which corresponds to the unit interval $[0, 1]$. We get $TRUTH(S_1)$ which consists of subobjects of S_1, which can be used as truth values for a fuzzy logic with values corresponding to intervals in $[0, 1]$.

In the context of denotators [15], we have an A-addressed truth denotator as an element of $A@TRUTH(I)$. Thus, we get the set $\mathbb{T}_I^A = A@TRUTH(I)$ of A-addressed truth denotators, and acquiring the total set of truth denotators is achieved by unioning over all $\mathbb{T}_I^A$ for every address A. This gives the set $\mathbb{T}_I$ of truth values.

Mazzola uses these ideas of truth value forms and truth denotators to define what he calls a *textual semiosis*, which is a language along with a semantics for the language. More specifically, a textual semiosis is defined as a set map

$$sig_{Den} : Tex \to Texig(Den)_I, \tag{6.13}$$

where $Tex \subset UNICODE$ is a subset of unicode strings, called *expressions*. The expressions are, intuitively, the statements of the language. To define $Texig(Den)_I$, first the category Den of denotators is to be defined. Then define "$Den_\infty = \coprod_{n\geq 1} Den^n$ to be the union of all powers of the arrow set ${}_1Den$, including the identities which represent the denotators as objects."[16, p. 406] Then we can define

$$Texig(Den)_I = \mathbb{T}_I^{Den_\infty}$$

to be the set of *characteristic functions* whose domain is the set of all finite length tuples of denotators or morphisms of denotators, and whose codomain is $\mathbb{T}_I$.

The map sig_{Den} is therefore simply a variation of the idea of categorical semantics, where the expressions of a language L are interpreted in some category $\mathscr{C}$ via a functor M. So sig_{Den} plays this role of M in categorical semantics, but sig_{Den} is a set map, rather than a functor.

Furthermore, it is demonstrated[4] that for every morphism of modules $h : I \to J$, there is a canonical form morphism

$$TRUTH(h) : TRUTH(I) \to TRUTH(J).$$

This allows one to define a morphism of textual semioses. For two textual semioses

$$sig^1_{Den} : Tex^1 \to Texig(Den)_I \text{ and } sig^2_{Den} : Tex^2 \to Texig(Den)_J$$

a morphism is a pair (u, h) where $u : Tex^1 \to Tex^2$ is a set map on expressions and $h : I \to J$

[4] [16, p. 409].

is a module morphism, such that the diagram

$$\begin{tikzcd} \mathit{Tex}^1 \arrow[r, "sig^1_{Den}"] \arrow[d, "u"'] & \mathit{Texig(Den)}_I \arrow[d, "\mathit{Texig(Den)}(h)"] \\ \mathit{Tex}^2 \arrow[r, "sig^2_{Den}"'] & \mathit{Texig(Den)}_J \end{tikzcd} \tag{6.14}$$

commutes. This thus defines the category of textual semioses.

We can of course translate all of these constructions by replacing Mazzola's forms with our structures. Also, for a set I (in **Rel**), we need not define $Val(I)$ (in **Str**) such that the functor $Fun(Val(I)) = @I$. Instead, we may take a subfunctor $Fu \rightarrowtail @I$. For instance, if we wish to define a topological structure on I, then we need to define a sieve $S \subset @I$ that generates this topological structure. Then we get

$$TRUTH(I) \underset{Fu \rightarrowtail @I}{\longrightarrow} \mathbf{Simple}(@I),$$

which can be used like in a sheaf topos, i.e., where the subobjects in $TRUTH(I)$ corresponding to open sets are used as truth values.

6.2.4 Abstract and Concrete Predicates

Summary. We provide a distinction between *abstract* and *concrete* predicates. We also take a step in defining formally Carnap's informal distinction between scientific and prescientific concepts.

— § —

In the traditional Tarskian setup, the semantics of an expression E in a language L is the extension of E given by some interpretation I. That is, the meaning of E is a set(-like entity) of individuals that satisfy E. While this is generally fine for mathematical objects, it is less convenient for languages that are not about the eternal objects one encounters in mathematics. For instance, if we define the property Bird, then this should apply not only to the birds that exist here and now, but to birds that have yet to be born or discovered. Clearly, the extension of Bird is constantly changing, as birds frequently die and are born. Therefore we should have a means of defining the semantics of a predicate independently of its extension. We call such 'pre-extensional' predicates *abstract predicates.* In essence, abstract predicates behave, as we will see, much like the extensional predicates one encounters in the traditional setup. However, the intuition is that the abstract predicates do not have extensions over domains of discourse that we wish to reason about, but over abstract domains of discourse, the individuals of which are used to *evaluate* the concrete individuals that we will wish to reason about. For instance, in the example from section 6.1.2 with temperatures, the temperature space $T \times T$ is used to establish a *criterion* for the relation Warmer. The subobject $Warmer_T \rightarrowtail T \times T$ consists of a comparison of *temperatures*, but not of the individuals that bear those temperatures. Thus, $Warmer_T$ serves as an abstract predicate, which will be actualized by a domain U of individuals once given a means of measurement $f : U \to T$.

Given such abstract relations that are 'pre-phenomenal' – in the sense that they do not apply to phenomana, but to spaces that interpret phenomena –, we are thus able to form abstract predicates in general, which would thus constitute an abstract language $\mathscr{L}_{\mathbb{P}}$. The language $\mathscr{L}_{\mathbb{P}}$ essentially sets up a *potential* world (hence the $\mathbb{P}$ subscript), which is *actualized* by giving concrete universes of individuals and rules for evaluating those individuals according to the abstract spaces that exist in $\mathscr{L}_{\mathbb{P}}$. Thus, the abstract predicates of $\mathscr{L}_{\mathbb{P}}$ can be used to actualize concrete predicates in a new language $\mathscr{L}_{A(\mathbb{P})}$ that extends $\mathscr{L}_{\mathbb{P}}$. As we will see, however, it is useful to specify what constitutes a permissible actualization of $\mathscr{L}_{\mathbb{P}}$, in order to refine the relationship between the potential and the actual. This is achieved by providing an *actualization condition*, to be discussed below.

6.2.4.1 Abstract Predicates

Summary. Abstract predicates are what enable us to define meaning independently of extensions. They provide a basis for concrete realizations of meaning.

— § —

We start with a definition of an *abstract predicate.*

Definition 6.6 (Abstract Predicate)**.** Let M be a structure, Ω^I a truth values object, and $C \rightarrowtail M$ the subobject of M given by the characteristic map $\chi_C : M \to \Omega^I$. An *abstract predicate* P is thus defined as the pair $P = (M, C)$. We call M the *evaluation space* of P and C the *criterion* of P.

We call M an *evaluation space* because it is the space that will be used to evaluate the elements of a 'concrete' universe of objects U. For instance, in the temperature example, the structure T, which has the structure of the field $\mathbb{R}$, is used to evaluate the temperatures of phenomena in some set U.

We call C a *criterion* because its elements are those in M that satisfy the abstract predicate P. Thus we know that, given a map $f : X \to M$ where X is a set of individuals, $f(x)$ satisfies the predicate if $\chi_C(f(x)) \neq \bot$.

Compound abstract predicates can be formed in the usual way, with logical connectives such as $\vee, \wedge, \implies$, and quantifiers $\forall, \exists$.

For a collection Π of abstract predicates, we can define a language $\mathscr{L}_{\mathbb{P}}$. To do this, recall from section 6.1.3 the definition of a signature.

To define $\mathscr{L}_{\mathbb{P}}$, we first define the signature $\Sigma_{\mathbb{P}} = (\mathfrak{M}, \varnothing, \mathfrak{C})$, where:

1. $\mathfrak{M}$ is the collection of sorts that will, upon interpretation, correspond to the evaluation spaces M of each abstract predicate P.

2. $\varnothing$ is the empty set.

3. $\mathfrak{C}$ is the collection of predicate symbols that will, upon interpretation, correspond to the criteria C of each abstract predicate P.

Notice that we do not provide any function symbols since we do not yet need to provide any mappings.

Defining a language $\mathscr{L}_{\mathbb{P}}$ over $\Sigma_{\mathbb{P}}$ would thus consist of specifying a set of first-order formulae over $\Sigma_{\mathbb{P}}$.

An interpretation I of $\Sigma_{\mathbb{P}}$ constitutes a functor $I : \Sigma_{\mathbb{P}} \to \mathbf{Str}$, or, converting **Str**-structures into **Set**-structures, a functor $I : \Sigma_{\mathbb{P}} \to \mathbf{Set}$ that sends each $\mathbf{M} \in \mathfrak{M}$ to an evaluation space M and each $\mathbf{C} \in \mathfrak{C}$ to a criterion C.

To understand how I interprets compound predicates, recall from [11, p. 811] that a *context* is a sequence $\Gamma = (x_1 : A_1, \ldots, x_n : A_n)$ of terms, each x_i of sort A_i. We say that a context Γ is *suitable* for a formula ϕ if all the free variables in ϕ occur in Γ. A formula-in-context $\Gamma.\phi$ is a formula ϕ with a specified context Γ. Thus, for a formula $\Gamma.\phi$ with context $\Gamma = (x_1 : A_1, \ldots, x_n : A_n)$, the interpretation of $\Gamma.\phi$ by I is a subobject

$$I(\Gamma.\phi) \rightarrowtail I(A_1, \ldots, A_n). \tag{6.15}$$

Thus, formulae over $\Sigma_{\mathbb{P}}$ constitutes what are called *abstract predicates*, and these abstract predicates are the expressions of a language $\mathscr{L}_{\mathbb{P}}$.

6.2.4.2 Actualizing Abstract Predicates

Summary. The method for actualizing abstract predicates is discussed. This is the process by which the pre-extensional abstract predicates are given extensional realizations.

— § —

To actualize abstract predicates, we first define an *actualization condition*:

Definition 6.7 (Actualization Condition)**.** Let $P = (M, C)$ be an abstract predicate (where M and C are objects in the semantic category). An *actualization condition* is a pair $K(P) = (\mathrm{V}_{\mathrm{dom}}, \mathrm{V}_{\mathrm{map}})$, where:

1. $\mathrm{V}_{\mathrm{dom}}$ is a predicate, used to define what constitutes a valid *domain* X for a map with codomain M.

2. $\mathrm{V}_{\mathrm{map}}$ is a predicate, use to define what constitutes a vaild *map* $f : X \to M$.

Since $\mathrm{V}_{\mathrm{map}}$ depends on $\mathrm{V}_{\mathrm{dom}}$, the format of the definition of $\mathrm{V}_{\mathrm{map}}$ is

$$\mathrm{V}_{\mathrm{map}}(f) \iff \mathrm{V}_{\mathrm{dom}}(\mathrm{dom}(f)) \wedge \phi, \tag{6.16}$$

where $\mathrm{dom}(f)$ sends f to its domain, and ϕ is a subformula that completes the definition. The idea is that $\mathrm{V}_{\mathrm{dom}}$ and $\mathrm{V}_{\mathrm{map}}$ provide descriptions of what constitutes a valid domain X and a valid map $f : X \to M$. For instance, the definition of $\mathrm{V}_{\mathrm{dom}}$ could be as simple as providing a choice of specified domains, such as

$$\mathrm{V}_{\mathrm{dom}}(X) \iff X = A \vee \cdots \vee X = N,$$

or it can be a more complex formula that picks out domains that satisfy more elaborate requirements.

The definition of $\mathrm{V}_{\mathrm{map}}$ will often be more complicated than that of $\mathrm{V}_{\mathrm{dom}}$, as the former will contain a description of the functional process itself that must be satisfied by a map $f : X \to M$. Usually, functions in a language are defined outside of the language, via definitional equalities such as

$$f(n) \equiv n + 1. \tag{6.17}$$

The definitional equality symbol $\equiv$ means that $f(n)$ is equal to $n+1$ *by definition.* This is in contrast to propositional equalities, such as e.g.

$$k(m + n) = km + kn, \tag{6.18}$$

since $k(m+n)$ is not *defined* as $km+kn$, but is a proposition – and thus requires proof – in a language L. Thus, propositional equalities are formulated *within* languages, whereas definitional equalities are meta-statements *about* languages. So we cannot provide definitional descriptions of functions in the definition of $\mathrm{V}_{\mathrm{map}}$, since $\mathrm{V}_{\mathrm{map}}$ is a compound predicate defined *within* the language. However, we can perform an *internalization* of definitional equalities into propositional equalities, thus bringing them into the language. For instance, $f(n)$ from Equation 6.2.4.2 can be brought into a language L via the formula

$$\forall n \in \mathbb{N}\big(f(n) = n + 1\big). \tag{6.19}$$

For the situation in which f does not have any clear rule, such as if f is a set map where everything is mapped in a haphazard way, then the internalization would be much more complex. For instance, we would have to describe f by specifying its value on every member of the domain, which would give a formula such as

$$f(n_1) = x_1 \wedge \cdots \wedge f(n_k) = x_k. \tag{6.20}$$

Given such disparities between the lengths of internalized function definitions, it is reasonable to establish a criterion for evaluating the descriptive complexity of a function. This is in the spirit of Kolmogorov complexity, which provides a complexity value to a computational object based on the length of the shortest computer program (in a specified programming language) that generates the object.

Analogously, given a function f, we define its *descriptive* complexity as so:

Definition 6.8 (Descriptive Complexity)**.** Let $f : X \to M$ be a map in some signature Σ. The *descriptive complexity* of f, denoted by $\delta(f)$, is defined as the length of the shortest formula over Σ that constitutes an internal definition of f. We denote the minimal formula of f by $\min_\phi(f)$, and thus $\delta(f) = \text{len}(\min_\phi(f))$.

Note that the length of a formula is the quantity of symbols used to construct it. So, for instance, the descriptive complexity of

$$\forall n \in \mathbb{N}\big(f(n) = n + 1\big)$$

is 14 (which includes brackets). Or, if we notated the operation + as a function on pairs, such as

$$\forall n \in \mathbb{N}\big(f(n) = +(n, 1)\big),$$

then the length is 17.

A function with no clear rule will have as its shortest internalization the formula of the form

$$f(n_1) = x_1 \wedge \cdots \wedge f(n_k) = x_k, \tag{6.21}$$

as defined in Equation 6.20. For any function f, denote by $\epsilon(f)$ its element-by-element description as given by formula 6.21. Since the length of each subexpression '$f(n_i) = x_i$' is 6, and we need an $\wedge$ for each member of the domain $n_i \in N$, except for one, then the length of $\epsilon(f)$ is

$$\text{len}(\epsilon(f)) = 6 \cdot card(N) + card(N) - 1. \tag{6.22}$$

So, for a function f, if $\text{len}(\min_\phi(f)) = \text{len}(\epsilon(f))$, then f is a *maximally complex* function.

On the other hand, a *minimally complex* function g would be any constant function

$$\forall x \in X\big(g(x) = c\big), \tag{6.23}$$

so long as $card(X) > 1$. The length of the minimal description of such a constant function is 11. Thus, for a finite domain X, we may classify functions h on X by descriptive complexity, with values ranging between 11 and $\text{len}(\epsilon(h))$. Therefore, descriptive complexity δ forms an equivalence relation (when applied to a set of functions).

In general, we can conceive of the descriptive complexity of a *predicate* as the minimal internal definition of that predicate. For instance, we could have the following definitions of $Q(x)$:

1. $Q(x) \iff x > 5$,
2. $Q(x) \iff x > 5 \wedge \neg(x = 2)$.

But $x > 5$ obviously implies $\neg(x = 2)$, so the first definition is sufficient.

Thus, for predicates V_{dom} and V_{map} that constitute an actualization condition for an abstract predicate P, we may evaluate them for their descriptive complexity.

We will return to the topic of descriptive complexity in Section 6.2.4.3 to see how it relates to scientific and prescientific concepts. But for now, let us return to the discussion of actualizing abstract predicates.

As defined above, for an abstract predicate $P = (M, C)$, an actualization condition is a pair $K(P) = (\mathrm{V}_{\mathrm{dom}}, \mathrm{V}_{\mathrm{map}})$. However, we need to define the domains over which an interpretation I of $\mathrm{V}_{\mathrm{dom}}$ and $\mathrm{V}_{\mathrm{map}}$ are to range. As a general convention – which need not be followed religiously – assume that $I(\mathrm{V}_{\mathrm{dom}})$ ranges over $\mathcal{U}_2$, and that $I(\mathrm{V}_{\mathrm{map}})$ ranges over the collection of set maps f whose domain is in $\mathcal{U}_2$ and whose codomain is in $\mathcal{U}_1$. Call this set of morphisms $F(\mathcal{U}_2, \mathcal{U}_1)$. With this information, we can define a special kind of extension[5] to the abstract signature $\Sigma_{\mathbb{P}}$. To derive this new signature, which we can call $\Sigma_{A_0(\mathbb{P})}$, we add the following to $\Sigma_{\mathbb{P}}$:

1. To $\Sigma_{\mathbb{P}}$-Sort, add the sorts $\mathbf{U}_2$ and $\mathbf{F}(\mathbf{U}_2, \mathbf{U}_1)$, that are to interpret to $\mathcal{U}_2$ and $F(\mathcal{U}_2, \mathcal{U}_1)$, respectively.

2. For a subset $\pi \subseteq \Pi$ of the abstract predicates that were used to construct $\mathscr{L}_{\mathbb{P}}$,[6] devise actualization conditions $K(p) = (\mathrm{V}_{\mathrm{dom}}(p), \mathrm{V}_{\mathrm{map}}(p))$ for $p \in \pi$, and for each $K(p)$ add therefore $\mathrm{V}_{\mathrm{dom}}(p)$ and $\mathrm{V}_{\mathrm{map}}(p)$ to Σ-Pred.

The resulting language $\mathscr{L}_{A_0(\mathbb{P})}$ establishes a first step toward actualizing the abstract language $\mathscr{L}_{\mathbb{P}}$, as we have now added some actualization conditions.

The next step, which leads to a concrete actualization, is then to specify the following: For a subset $\rho \subseteq \pi$ of actualization conditions from π, we need to add the following to $\Sigma_{A_0(\mathbb{P})}$ for *each* actualization condition $K(r) \in \rho$:

1. A set $\mathbf{D}_r$ of sorts to $\Sigma_{A_0(\mathbb{P})}$-Sort. Each sort $\mathbf{d} \in \mathbf{D}_r$ will interpret to an element in the interpretation of $\mathrm{V}_{\mathrm{dom}}(r)$. That is, for an interpretation I, we will have that $I(\mathbf{d}) \in I(\mathrm{V}_{\mathrm{dom}}(r))$. This is because $I(\mathrm{V}_{\mathrm{dom}}(r))$ will be the collection of valid domains of objects that can map into the evaluation space M of some abstract predicate P_r, and thus $I(\mathbf{d})$ will be one of these domains of objects.

2. A set Φ_r of function symbols to $\Sigma_{A_0(\mathbb{P})}$-Fun. Each function symbol $\varphi \in \Phi_r$ is defined on a respective sort $\mathbf{d} \in \mathbf{D}_r$. We also have that $I(\varphi) \in I(\mathrm{V}_{\mathrm{map}}(r))$. This is because $I(\mathrm{V}_{\mathrm{map}}(r))$ will be the collection of valid maps into the evaluation space M of some abstract predicate P_r, and thus $I(\varphi)$ will be one of these maps.

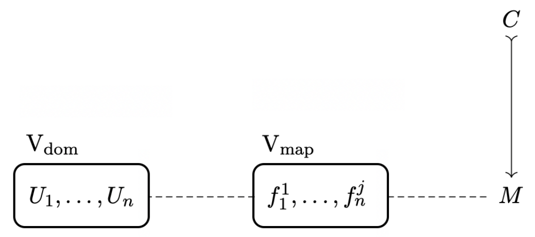
We can call such an extension $\Sigma_{A(\mathbb{P})}$. The subscript $A(\mathbb{P})$ is meant to suggest that $\Sigma_{A(\mathbb{P})}$ actualizes (some of) the potentialities – i.e., abstract predicates – in $\Sigma_{\mathbb{P}}$. Figure 6.4 demonstrates the contrasts between $\Sigma_{\mathbb{P}}$, $\Sigma_{A_0(\mathbb{P})}$, and $\Sigma_{A(\mathbb{P})}$.

Given this scheme for dealing with the ascent from abstract to concrete, we can construct a partial order over the 'base' signature $\Sigma_{\mathbb{P}}$. Since in moving from abstract predicates,

[5]We mean 'extension' in the sense of extending a language; we do not mean denotational extension.
[6]See the definition of $\Sigma_{\mathbb{P}}$ in Section 6.2.4.1.

$$\begin{tikzcd}
C \arrow[d, tail] \\
M
\end{tikzcd}$$

(a) $\mathscr{L}_{\mathbb{P}}$ consists of abstract predicates such as these.



(b) $\Sigma_{A_0(\mathbb{P})}$ adds predicates for valid domains and valid maps, but does not yet specify these maps into M. They are still merely 'potential'.

$$\begin{tikzcd}
 & C \arrow[dd, tail] \\
U_1 \arrow[dr, bend left, "f_{11}"] \arrow[dr, bend right, "f_{1k_1}"'] & \\
\vdots & M \\
U_n \arrow[ur, bend left, "f_{n1}"] \arrow[ur, bend right, "f_{nk_n}"'] &
\end{tikzcd}$$

(c) $\Sigma_{A(\mathbb{P})}$ realizes some of the maps that are specified by $\Sigma_{A_0(\mathbb{P})}$.

Figure 6.4

to actualization conditions, to actualizations, we are adding to the components of $\Sigma_{\mathbb{P}}$, the partial order is defined component-wise as so: We say that $\Sigma_\zeta \preceq \Sigma_\psi$ if all of the following conditions hold:

1. Σ_ζ-Sort $\subseteq \Sigma_\psi$-Sort,
2. Σ_ζ-Fun $\subseteq \Sigma_\psi$-Fun,
3. Σ_ζ-Pred $\subseteq \Sigma_\psi$-Pred.

Thus we get the partial order $\mathbf{P}\Sigma_{\mathbb{P}}$ given by such extensions, and paths through $\mathbf{P}\Sigma_{\mathbb{P}}$ correspond to increases of actualized potentialities.

6.2.4.3 Scientific and Prescientific Concepts

Summary. We elaborate upon Carnap's distinction between scientific and prescientific concepts. We provide a criterion that is more general than Carnap's criterion for scientific concepts.

— § —

Let us now return to actualization conditions. For an abstract predicate $P = (M, C)$, we give an actualization condition as a pair of predicates $K = (\mathrm{V}_{\mathrm{dom}}, \mathrm{V}_{\mathrm{map}})$. Furthermore, we can provide a measure of the descriptive complexities of $\mathrm{V}_{\mathrm{dom}}$ and $\mathrm{V}_{\mathrm{map}}$, respectively. Returning to the discussion on explication in Section 6.1.2, in the discussion of the abstract predicate Warmer, we had two versions, with respective evaluation spaces S and T. S consists of bodily sensations, whereas T consists of real numbers. Furthermore, S is considered a naive evaluation space, whereas T is considered more scientific and precise. Why is this? We may identify some of the aspects of T, such as the following:

- It is a 'large' space, as it contains infinite elements.
- There is a precise means for differentiating between the elements in T, as a result of its algebraic structure. For instance, for two temperatures $s, t \in T$, we can evaluate their difference $|t - s|$. Moreover, since the space is infinite, and moreover a continuum, we have very fine-grained differentiation between the elements. Thus we can make very precise measurements, and also differentiate between such measurements.
- T has a rich structure, which is that of a field. On the other hand, S is simply a set, with no clear structure.
- We may unambiguously refer to the elements of T. This is in contrast to the elements of S, which are not clearly understood, and thus there is no clear understanding of what is *actually* being denoted when we refer to an element $s \in S$. Furthermore,

when we refer to an element $t \in T$, there is no 'extra' reference. When I refer to a real number $t \in T$, there is no extra information that I need to understand t, as its nature is determined completely by its role in the structure of T. On the other hand, given the set S of bodily sensations, and a sensation $s \in S$, then this suggests something external to S itself, namely, the actual structure of the sensations themselves.

These are a few points that may aid in the understanding of what constitutes a scientific evaluation space M, although by no means do we have a scientific criterion for what constitutes a scientific evaluation space.

Now, suppose we have an abstract predicate $P = (M, C)$. We have the following

Principle 6.1. *An actualization condition $K = (\mathrm{V}_{\mathrm{dom}}, \mathrm{V}_{\mathrm{map}})$ for P is more scientific the lower is the descriptive complexity of* $\mathrm{V}_{\mathrm{dom}}$ *and* $\mathrm{V}_{\mathrm{map}}$.

We make this claim because it demonstrates the following:

1. We have *exact* and *simple* knowledge of the *domains of objects* that the concept can apply to. Our knowledge of domains is *exact* because we have $\mathrm{V}_{\mathrm{dom}}$ defined formally, and it is *simple* because $\mathrm{V}_{\mathrm{dom}}$ has low descriptive complexity.

2. We have *exact* and *simple* knowledge of the *rules* that evaluate the domains of objects given by $\mathrm{V}_{\mathrm{dom}}$. Again, our knowledge of rules is *exact* because we have $\mathrm{V}_{\mathrm{map}}$ defined formally, and it is *simple* because $\mathrm{V}_{\mathrm{map}}$ has low descriptive complexity.

Recall that Carnap's four criteria for explication are (1) similarity to the explicandum, (2) exactness, (3) fruitfulness, (4) simplicity. Thus, we have fulfilled criteria (2) and (4). It would be much more difficult, however, to formalize what constitutes criteria (1) and (3). However, we might speculate that (1) could be achieved if one were to define some sort of similarity relation between the expressions of the language of the explicandum and the expressions of the language of the explicatum. On the other hand, we may be able to formulate (3) via some function that maps expressions in the explicatum language to their 'degree of fruitfulness'. However, such formal devices for determining the satisfaction of (1) and (3) may not always be useful, whereas formalization of (2) and (4) has direct application, and I claim is relevant to scientific practice.

Bibliography

[1] Aristotle. “Metaphysics”. In: *The complete works of Aristotle: the revised Oxford translation.* Princeton University Press, 1991.

[2] Aristotle. “Physics”. In: *The complete works of Aristotle: the revised Oxford translation.* Princeton University Press, 1991.

[3] Aristotle. *The complete works of Aristotle: the revised Oxford translation.* eng. Princeton, N.J: Princeton University Press, 1991.

[4] Robert Brandom. *Making it explicit: Reasoning, representing, and discursive commitment.* Harvard university press, 1994.

[5] Rudolf Carnap. *Logical foundations of probability.* University of Chicago press, 1950.

[6] Brendan Fong and David I Spivak. *An invitation to applied category theory: seven sketches in compositionality.* Cambridge University Press, 2019.

[7] Gottlob Frege. *The foundations of arithmetic: A logico-mathematical enquiry into the concept of number.* Northwestern University Press, 1980.

[8] Jean-Yves Girard. “From foundations to ludics”. In: *Bulletin of Symbolic Logic* 9.2 (2003), pp. 131–168.

[9] Paul Guyer and Allen W Wood. *Critique of pure reason.* 1998.

[10] Wilfrid Hodges et al. *A shorter model theory.* Cambridge university press, 1997.

[11] Peter T Johnstone. *Sketches of an Elephant: 2 Volume Set.* 2002.

[12] Alain Lecomte. *Meaning, logic and ludics.* World Scientific, 2011.

[13] Saunders Mac Lane. *Categories for the working mathematician.* Vol. 5. Springer Science & Business Media, 2013.

[14] Saunders MacLane and Ieke Moerdijk. *Sheaves in geometry and logic: A first introduction to topos theory.* Springer Science & Business Media, 2012.

[15] Guerino Mazzola. “Denotators”. In: *The Topos of Music I: Theory.* Springer, 2017, pp. 41–85.

[16] Guerino Mazzola. “Predicates”. In: *The Topos of Music I: Theory.* Springer, 2017, pp. 327–349.

[17] Guerino Mazzola. *The topos of music: geometric logic of concepts, theory, and performance*. Birkhäuser, 2012.

[18] Jaroslav Peregrin. “Inferentialism and normativity”. In: (2013).

[19] Wilfrid Sellars. “Inference and meaning”. In: *Mind* 62.247 (1953), pp. 313–338.

[20] Wilfrid Sellars. “Language as Thought and as Communication”. In: *Philosophy and Phenomenological Research* 29.4 (1969), pp. 506–527.

[21] Wilfrid Sellars. *Language, Rules and Behavior', S. Hook (ed.): John Dewey: Philosopher of Science and Freedom*. 1949.

[22] Ludwig Wittgenstein. *Philosophical investigations*. John Wiley & Sons, 2010.