A Category-theoretical Meta-analysis of Definitions of Disentanglement

Yivan Zhang [1 2] **Masashi Sugiyama** [2 1]

Abstract

Disentangling the factors of variation in data is a fundamental concept in machine learning and has been studied in various ways by different researchers, leading to a multitude of definitions. Despite the numerous empirical studies, more theoretical research is needed to fully understand the defining properties of disentanglement and how different definitions relate to each other. This paper presents a meta-analysis of existing definitions of disentanglement, using category theory as a unifying and rigorous framework. We propose that the concepts of the cartesian and monoidal products should serve as the core of disentanglement. With these core concepts, we show the similarities and crucial differences in dealing with (i) functions, (ii) equivariant maps, (iii) relations, and (iv) stochastic maps. Overall, our meta-analysis deepens our understanding of disentanglement and its various formulations and can help researchers navigate different definitions and choose the most appropriate one for their specific context.

1. Introduction

Disentanglement, in machine learning, refers to the ability to identify and separate the underlying factors that contribute to a particular variation in data (Bengio et al., 2013). It is a process of breaking down a complex phenomenon into simpler components. It has been suggested that disentangled representation learning is a promising way toward reliable, interpretable, and data-efficient machine learning (Locatello et al., 2019; Montero et al., 2020; Dittadi et al., 2021).

Because disentanglement is an important concept, many researchers have approached this problem from different angles, resulting in various definitions, metrics, methods, and models. Some definitions are based on the intuition that: (1. modularity) a change in one factor should lead to a change in a single code; (2. compactness/completeness) a factor should be associated with only one code; and (3. explicitness/informativeness) the code should be able to predict the factor (Ridgeway & Mozer, 2018; Eastwood & Williams, 2018). Another line of research is based on group theory and representation theory (Cohen & Welling, 2014; 2015; Higgins et al., 2018), where the mapping from the data to the code is required to be equivariant to product group actions, preserving the product structure of automorphisms (a.k.a. symmetries). Meanwhile, information theory (Chen et al., 2018) and invariance (Higgins et al., 2017) also play an important role in characterizing disentanglement.

Then why do we want to conduct a *meta-analysis*? Because we study the theories and techniques of disentanglement, yet our definitions of it are quite *entangled*. Although large-scale experimental studies exist (Locatello et al., 2019), theoretical analyses and systematic comparisons are limited (Sepliarskaia et al., 2019; Carbonneau et al., 2022). Several important questions remain to be answered:

- What are the defining properties of disentanglement?
- What operations and structures are essential, and what are specific to the task?
- Given two definitions or metrics, does one imply the other in any situation?
- Are the existing algebraic and statistical approaches compatible with one another?

Things quickly become complicated without an abstract language to describe existing results.

Category theory (Borceux, 1994; Awodey, 2006; Leinster, 2014) is particularly suitable for designing and organizing a system of this level of complexity. It has found applications in many scientific fields (Baez, 2017; Bradley, 2018; Fong & Spivak, 2019), recently also in machine learning (Gavranović, 2019; de Haan et al., 2020; Shiebler et al., 2021; Dudzik & Veličković, 2022). In this work, we aim to *disentangle the definitions of disentanglement* from a categorical perspective.

In Section 2, we first introduce the essential concepts of the *cartesian product* and *monoidal product*, which we argue should be the core of disentanglement. Next, we look into

[1]The University of Tokyo, Tokyo, Japan [2]RIKEN AIP, Tokyo, Japan. Correspondence to: Yivan Zhang <yivanzhang@ms.k.u-tokyo.ac.jp>.

Proceedings of the 40^{th} International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023. Copyright 2023 by the author(s).

the requirements based on examples and counterexamples through Sections 3 to 6. We use the categories of (1. $\mathbf{Set}$) sets and functions to define the concepts of modularity and explicitness as the defining properties of disentanglement (Ridgeway & Mozer, 2018); (2. $[\mathbf{S}, \mathbf{Set}]$) functors and natural transformations to generalize to actions of an algebra (monoid, group, etc.) and equivariant maps (Higgins et al., 2018); (3. $\mathbf{Rel}$) sets and relations as an example of a symmetric monoidal category; and (4. $\mathbf{Stoch}$) measurable spaces and stochastic maps to introduce the concept of the Markov category (Fritz, 2020) and explain how we should use the copy/delete/projection operations to characterize disentanglement. A full-blown example is given in the end.

It is worth clarifying that this paper does *not* discuss metrics, models, methods, supervision, and learnability. Also, our contribution is *not* to category theory itself, as the math we used is not new. However, our work shows how category theory can transfer and integrate knowledge across disciplines and how abstract definitions can simplify a complex system (Baez, 2017). We hope our work is an initial step toward a full understanding of disentanglement.

2. Product: Core of Disentanglement

In this section, we briefly review two important categorical concepts — the *cartesian product* and *monoidal product*, which are the core of the disentanglement. We will omit many basic concepts such as the category, functor, natural transformation, and monad. Note that we frequently use *commutative diagrams* (Awodey, 2006) and *string diagrams* (Selinger, 2010) as graphical calculus (See Appendix A.1).

2.1. Cartesian Category

Let us dive into the definition of the (cartesian) product:

Definition 1 (Product)**.** In any *category* $\mathbf{C}$, a *product* of two *objects* A and B is an object $A \times B$, together with two *morphisms* $A \xleftarrow{p_1} A \times B \xrightarrow{p_2} B$, called *projections*, satisfying the *universal property*:

$$\begin{array}{ccccc} & & C & & \\ & \swarrow^{f_1} & \downarrow{\scriptstyle \langle f_1, f_2 \rangle} & \searrow^{f_2} & \\ A & \xleftarrow{p_1} & A \times B & \xrightarrow{p_2} & B \end{array} \tag{1}$$

Given any object C and morphisms $A \xleftarrow{f_1} C \xrightarrow{f_2} B$, there exists a *unique* morphism $\langle f_1, f_2 \rangle : C \to A \times B$, called a *paring* of f_1 and f_2, such that $f_1 = p_1 \circ \langle f_1, f_2 \rangle$ and $f_2 = p_2 \circ \langle f_1, f_2 \rangle$.

The gist is that any morphism $C \xrightarrow{f} A \times B$ *to* a product is merely a pair of component morphisms $A \xleftarrow{f_1} C \xrightarrow{f_2} B$, and all such morphisms arise this way. However, note that a morphism $A \times B \to C$ *from* a product can depend on both components.

We will be needing the following definitions and properties:

- The *product morphism* of $f : A \to C$ and $g : B \to D$ is defined as $f \times g : A \times B \to C \times D := \langle f \circ p_1, g \circ p_2 \rangle$, which makes product $\times : \mathbf{C} \times \mathbf{C} \to \mathbf{C}$ a *bifunctor*.
- The *diagonal morphism* of an object A is defined as $\Delta_A : A \to A \times A := \langle \mathrm{id}_A, \mathrm{id}_A \rangle$, which "*duplicates*" A.
- The *terminal object* 1, if exists, is the *unit* of the product: for any object A, there is a *unique terminal morphism* $e_A : A \to 1$, which "*deletes*" A, and $A \times 1 \cong A \cong 1 \times A$.
- The product is *associative* up to isomorphism $\alpha_{A,B,C} : (A \times B) \times C \cong A \times (B \times C) := \langle p_1 \circ p_1, p_2 \times \mathrm{id}_C \rangle$, which allows us to define products $\prod_{i=1}^N A_i = A_1 \times \cdots \times A_N$ and projections $p_i : \prod_{i=1}^N A_i \to A_i$ for $N \geq 2$ objects. We use subscript $f_i := p_i \circ f$ as an abbreviation.
- The product is *commutative* up to isomorphism $\beta_{A,B} : A \times B \cong B \times A := \langle p_2, p_1 \rangle$.

A *cartesian category* is a category with all finite products, i.e., all binary products and a terminal object.

2.2. Monoidal Category

Having all products is sometimes too strong a condition. Besides, the product, if exists, is not always an appropriate concept for disentanglement. Therefore, sometimes we need to consider a weaker notion of the "product":

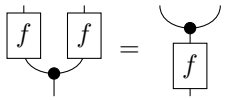
Definition 2 (Symmetric monoidal category)**.** A *symmetric monoidal category* $(\mathbf{C}, \otimes, I)$ is a category $\mathbf{C}$ equipped with a *monoidal product* $\otimes : \mathbf{C} \times \mathbf{C} \to \mathbf{C}$ and a *monoidal unit* I, which is unital, associative, and commutative up to natural isomorphisms and subject to some coherence conditions.

The monoidal products are weaker because they do not need to satisfy the universal property, so there are no canonical projections anymore. A *cartesian (monoidal) category* is a symmetric monoidal category whose monoidal product is given by the cartesian product. However, many interesting monoidal categories are not cartesian.

Some symmetric monoidal categories have extra structures or properties, including

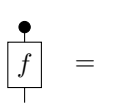
- *monoidal category with diagonals* $\Delta_A : A \to A \otimes A$, which is natural in A if

$$\begin{array}{ccc} A & \xrightarrow{f} & B \\ {\scriptstyle \Delta_A}\downarrow & & \downarrow{\scriptstyle \Delta_B} \\ A \otimes A & \xrightarrow{f \otimes f} & B \otimes B \end{array} \qquad \text{(2)}$$



- *semicartesian (monoidal) category*, whose monoidal unit I is a terminal object:

$$\begin{array}{ccc} A & \xrightarrow{f} & B \\ & {\scriptstyle e_A} \searrow \quad \swarrow {\scriptstyle e_B} & \\ & I & \end{array} \qquad \text{(3)}$$



- *monoidal category with projections* $\pi_1 : A \otimes B \to A$ and $\pi_2 : A \otimes B \to B$ (Franz, 2002; Leinster, 2016), and
- *Markov category* (Fritz, 2020, Definition 2.1).

They have the following relationship:

$$\begin{array}{ccccc} \text{cartesian} & \subset & \text{Markov} & \subset & \text{semicartesian} \\ \cap & & & & \| \\ \text{diagonals} & \subset & \text{monoidal} & \supset & \text{projections} \end{array} \tag{4}$$

These structures and properties will be important in the rest of this paper.

3. Sets and Functions

Equipped with these concepts, let us now look at the definitions of disentanglement. **Set**, the category of sets and functions, serves as our primary example. **Set** is cartesian, whose product is given by the *Cartesian product* of sets.

We use $[1..N]$ to denote the set of numbers from 1 to N. We use $\backslash i$ as an abbreviation of $[1..N] \setminus \{i\}$, i.e., the set of numbers from 1 to N except i.

3.1. Generating Process

First, let us consider how the data is generated from a set of factors. If all combinations of factors are equally possible (cf. Section 5), we can assume that

Assumption 1. The set of *factors* $Y := \prod_{i=1}^{N} Y_i$ is a *product* of N sets.

Then, let X be the set of *observations*. A *generating process* $g : Y \to X$ is simply *a morphism from a product*, i.e., a function with multiple inputs. It is an "*entangling process*" because we do not have any structural assumptions on X. However, we need some basic requirements for g to ensure that the analysis is meaningful. For starters, we assume that

Assumption 2. $g : Y \rightarrowtail X$ is a *monomorphism*.

This means that if two observations are the same, their underlying factors must be the same, too. This assumption avoids the model not satisfying a disentanglement definition simply because of a wrong choice of factors.

3.2. Encoding Process

Next, we consider how an *encoding process* $f : X \to Z$ can exhibit disentanglement and what desiderata are. Following Ridgeway & Mozer (2018) and Eastwood & Williams (2018), we call Z the set of *codes*, which should also be a product. In this work, we consider a simple case where

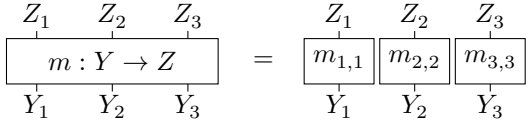
Assumption 3. The codes Z also have N components, and the code projections $p_i : Z \to Z_i$ are known a priori.

Based on Assumption 3, we present our first definition:

> **Disentanglement 1** (A morphism to a product)**.** In a category $\mathbf{C}$, a disentangled encoding process is a morphism $f : X \to Z$ to a *product* $Z := \prod_{i=1}^{N} Z_i$.

This is perhaps the minimal requirement for an encoder to exhibit some level of disentanglement. It means that the encoder outputs multiple components, and we can extract each component without losing any information. Note that **D**. 1 does not even rely on the ground-truth factors Y and a generating process g.[1]

Let us now improve **D**. 1. A disentanglement requirement that many researchers agree on is *modularity*, such that "*each code conveys information about at most one factor*" (Ridgeway & Mozer, 2018). It is natural to consider the composition $m : Y \to Z := f \circ g$ of a generating process g and an encoding process f, which we call a *code generating process* (w.r.t. a given encoding f), while $g : Y \rightarrowtail X$ can be referred to as a *data generating process*. Then, modularity is a property of a code generating process:

> **Disentanglement 1.1.** $m = \prod_{i=1}^{N}(m_{i,i} : Y_i \to Z_i)$.
>
> 
>
>
> "*The i-th code only encodes the i-th factor.*"

Morphisms m, m_i, and $m_{i,i}$ have the following relationship:

Proposition 1. $\forall i \in [1..N].\ m_i := p_i \circ m = m_{i,i} \circ p_i$.

$$\begin{array}{ccc} Y & \xrightarrow{m} & Z \\ {\scriptstyle p_i}\downarrow & \searrow^{m_i} & \downarrow{\scriptstyle p_i} \\ Y_i & \xrightarrow[m_{i,i}]{} & Z_i \end{array} \tag{5}$$

D. 1.1 is straightforward and intuitive, but there is one difficulty: it relies on the *existence* of some other morphisms. Given m, verifying if $m_{i,i}$ exists is not trivial. Although, if **D**. 1.1 holds, we can construct $m_{i,i}$ from m as follows:

Proposition 2. $\forall i \in [1..N].\ \forall y_i : 1 \to Y_i.\ m_{i,i} = Y_i \xrightarrow{\cong} 1 \times \cdots \times Y_i \times \cdots \times 1 \xrightarrow{y_1 \times \cdots \times \mathrm{id}_{Y_i} \times \cdots \times y_N} Y \xrightarrow{m} Z \xrightarrow{p_i} Z_i$.

In words, we can choose other factors arbitrarily, and a modular encoder should give us the same code. This inspires us to have a more *verifiable* definition as follows.

A good property of **Set** is that it is *cartesian closed*, i.e., it has *exponential objects*, given by the sets of functions. Let $\widehat{m_i} : Y_{\backslash i} \to {Z_i}^{Y_i}$ be the *exponential transpose* (*currying*) of $m_i : Y \to Z_i$. To check modularity, we can verify if

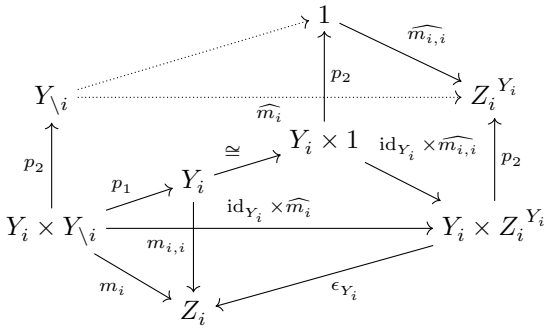
> **Disentanglement 1.2.** $\widehat{m_i}$ is a *constant morphism*.

Therefore, we can obtain the exponential transpose first and check whether it is constant. Even better, we can guarantee that these definitions are equivalent:

Theorem 3. ***D***. 1.1 $\leftrightarrow$ ***D***. 1.2.

[1] **D**. 1 refers to **Disentanglement** 1.

Proof. Diagram chase.



(6)

□

Up to this point, we defined modularity in a cartesian closed category like **Set**. However, we point out that modularity alone is not sufficient:

Example (Constant)**.** Let Z be the terminal object $1^N \cong 1$. The terminal morphism $e_Y : Y \to 1$ satisfies **D**. 1.1.

That is, *an encoder sending everything to singletons is perfectly modular but also completely useless*. Therefore, in addition to modularity, we should measure how useful and informative the codes are.

3.3. Decoding Process

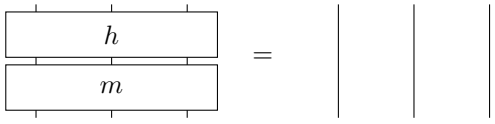
This is where the concepts of *explicitness* (Ridgeway & Mozer, 2018) or *informativeness* (Eastwood & Williams, 2018) come in, meaning that "*the factors can be precisely determined from the codes*". It might be tempting to define explicitness as

Disentanglement 1.3. f is an *inverse* of g.

Then, the factors can be completely reconstructed from the observations. A drawback of **D**. 1.3 is that it requires the code set Z to be the same as the factor set Y, so Y needs to be known during training. However, it is common that an encoder $f : X \to Z$ is trained with self-supervision or weak supervision (Shu et al., 2020; Wang et al., 2021), and the ground-truth factors Y are only available during evaluation.

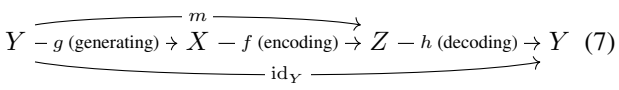
Therefore, we weaken the requirement and define the explicitness of a code generating process as

Disentanglement 1.4. m is a *split monomorphism*.



"*The codes encode the factors faithfully.*"

This means that there exists a morphism $h : Z \to Y$, which we call a *decoding process*, such that $h \circ m = \mathrm{id}_Y$. In other words, h is a *retraction* of m. To summarize, we will focus on the following morphisms from now:



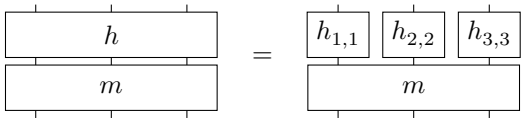
(7)

Note that explicitness only indicates if the factors can be recovered. We may end up with entangled codes:

Example (Rotation)**.** Let Y be a vector space. A rotation is an invertible linear transformation and satisfies **D**. 1.4.

To avoid this, we may want the decoder to be modular, too. This property is related to the concepts of *compactness* (Ridgeway & Mozer, 2018) and *completeness* (Eastwood & Williams, 2018), meaning that "*a factor is associated with only one code*" (See also Appendix A.2). Like **D**. 1.1, we can require h to be a product morphism:

Disentanglement 1.5. $h = \prod_{i=1}^{N}(h_{i,i} : Z_i \to Y_i)$.



"*The i-th code encodes the i-th factor faithfully.*"

If an encoder has a modular decoder, we can safely drop other codes if a downstream task only relies on a subset of factors. For example, if a task only depends on factor Y_i, then a component encoder $f_i : X \to Z_i$ can encode sufficient information for this task.

We point out that an encoder with a modular decoder may not be modular itself:

Example (Duplicate)**.** Let Z be $Y \times Y$. The diagonal morphism Δ_Y satisfies **D**. 1.5 with a retraction $p_1 \times p_2$.

This means that a *non-modular* and explicit encoder may copy all the factors for each code $Z_i := Y$, and its modular decoder $h_{i,i} : Z_i \to Y_i := p_i$ can simply project the code to each component, which is not what we expect.

A potential remedy to this issue is to require that the code does *not* contain any other information except for the target factor:

Disentanglement 1.6. $\forall i, j \in [1..N]. \nexists h_{i,j} : Z_i \to Y_j. (i \neq j) \wedge (h_{i,j} \circ m_i = p_j)$.

"*The i-th code does not encode the j-th factor.*"

However, **D**. 1.6 is even harder to verify than **D**. 1.1 because it relies on the *non-existence* of some morphisms. This is another difficulty in dealing with non-modular encoders.

Fortunately, we can guarantee that a *modular* and explicit encoder must have a modular decoder:

Theorem 4. (***D***. 1.1 $\wedge$ ***D***. 1.4) $\to$ ***D***. 1.5.

It is now clear that modularity (**D**. 1.1) and explicitness (**D**. 1.4) of an encoder should be the defining properties of disentanglement and our main focus when designing and evaluating disentangled representation learning algorithms. Waiving either of these requirements could cause problems. Our analysis supports similar arguments made by Ridgeway & Mozer (2018), Duan et al. (2020), and Carbonneau et al. (2022).

A minor issue is that a modular and explicit encoder may have a "*non-explicit*" decoder:

Example (Redundancy)**.** Let Z be $(Y_1 \times Y_1) \times Y_2$. The morphism $m = \Delta_{Y_1} \times \mathrm{id}_{Y_2}$ satisfies both **D**. 1.1 and **D**. 1.4.

It means that $Z_1 := Y_1 \times Y_1$ contains redundant information of Y_1. All meaningful codes are of the form $((y_1, y_1), y_2)$, while codes of the form $((y_1, y_1'), y_2)$ are meaningless and should not be decoded. In categorical terms, m is a product morphism, a split monomorphism, but not an *epimorphism*. If we want to *traverse the code space*, we can additionally require m to be a (split) epimorphism.

4. Algebra Actions and Equivariant Maps

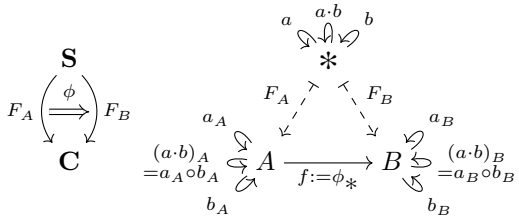
We can simply change the category from $\mathbf{Set}$ *to* $[\mathbf{S}, \mathbf{Set}]$.

In this section, we explain the above sentence by showing three ways to extend **D**. 1 and how it relates to the definition based on the direct product of groups (Higgins et al., 2018).

$[\mathbf{S}, \mathbf{C}]$ denotes the *functor category* of *functors* from $\mathbf{S}$ to $\mathbf{C}$ and *natural transformations* between these functors. We call the category $\mathbf{S}$ a *scheme*. To see how it relates to the existing algebraic formulation of disentanglement, we need the following well-known fact:

Definition 3 (Equivariance as naturality)**.** Many algebraic structures, such as monoids and groups, can be considered as *single-object categories*. Then, an *action* of an algebra at an object A is precisely a *functor* $F_A : \mathbf{S} \to \mathbf{C}$ from the corresponding scheme $\mathbf{S}$ to a category $\mathbf{C}$ containing A, and an *equivariant map* $f : A \to B$ between two actions F_A and F_B is precisely a *natural transformation* $\phi : F_A \Rightarrow F_B$.

An example is shown below:



(8)

We use subscript $a_A := F_A a$ as an abbreviation. We can see that F_A and F_B send the single $\mathbf{S}$-object $*$ to $\mathbf{C}$-objects A and B and send endomorphisms to endomorphisms. In this way, we can consider $\mathbf{S}$ as *syntax* and $\mathbf{C}$ as *semantics*.

Example (Regression vs. Ranking)**.** Not all problems can be formulated using only endomorphisms, let alone groups. Some *ranking* problems (Liu, 2011) roughly correspond to finding order-preserving functions, which is equivariant to actions of the free monoid of natural numbers $\mathbb{N}$. However, the usual *regression* problems also require the preservation $f(x_0) = 0$ of the zero point, which is a nullary operation $\mathrm{zero} : 1 \to \mathbb{N}$ (a morphism from a singleton to the set $\mathbb{N}$).

4.1. Product Category and Functor Product

Let us now consider the products of categories and functors. We highlight the following two important properties:

- The category of small categories $\mathbf{Cat}$ is cartesian closed, with the product and exponential object given by the *product category* $\mathbf{S}_1 \times \mathbf{S}_2$ and functor category $[\mathbf{S}, \mathbf{C}]$.
- If $\mathbf{C}$ has *(co)limits* of a certain shape (e.g., product), then $[\mathbf{S}, \mathbf{C}]$ has pointwise (co)limits of the same shape (e.g., *functor product* $F_1 \times^{\mathbf{S}} F_2 : \mathbf{S} \xrightarrow{\langle F_1, F_2 \rangle} \mathbf{C} \times \mathbf{C} \xrightarrow{\times} \mathbf{C}$).[2]

Knowing if $\mathbf{C}$ has products then so does $[\mathbf{S}, \mathbf{C}]$, we can now extend **D**. 1 straightforwardly by *simply changing the category from* $\mathbf{C}$ *to* $[\mathbf{S}, \mathbf{C}]$:

Disentanglement 2 (A natural transformation to a functor product)**.** Let $\mathbf{S}$ be a category, $\mathbf{C}$ be a category with products, and $F_X, F_{Z_i} : \mathbf{S} \to \mathbf{C}, i \in [1..N]$ be functors. A disentangled encoding process is a morphism to a product in $[\mathbf{S}, \mathbf{C}]$, i.e., a natural transformation $\phi : F_X \Rightarrow F_Z$ to a *functor product* $F_Z := \prod_{i=1}^{N} F_{Z_i}$.

In other words, the same scheme $\mathbf{S}$ has N different models via F_{Z_i} in $\mathbf{C}$, which are combined into a single model via product F_Z. In the product group action example (Higgins et al., 2018), **D**. 2 means that the product group is viewed as a single-object category $\mathbf{S}$, and the product structure of automorphisms is preserved via the functor product.

Another approach is to view each group as a single-object category and the product group as a product category. Then, we can use the following definition:

Disentanglement 3 (A natural transformation between multifunctors)**.** Let $\mathbf{S} = \prod_{i=1}^{N} \mathbf{S}_i$ be a product category, $\mathbf{C}$ be a category, and $F_X, F_Z : \mathbf{S} \to \mathbf{C}$ be multifunctors. A disentangled encoding process is a morphism in $[\mathbf{S}, \mathbf{C}]$, i.e., a natural transformation $\phi : F_X \Rightarrow F_Z$ between *multifunctors*.

That is, a scheme $\mathbf{S}$ with N components has a model in $\mathbf{C}$. We can see that **D**. 2 defines disentanglement via the *product of functors* (based on the product in the codomain category $\mathbf{C}$), while **D**. 3 uses the *product of domain categories* (based

[2] We reserve the term *product functor* to the product morphism in $\mathbf{Cat}$, i.e., $F_1 \times F_2 : \mathbf{S}_1 \times \mathbf{S}_2 \to \mathbf{C}_1 \times \mathbf{C}_2$, to avoid confusion.

on the product in **Cat**). They have their own application scenarios, but due to space limits, we will not study **D**. 2 and **D**. 3 further in this paper.

4.2. Product-preserving Functors

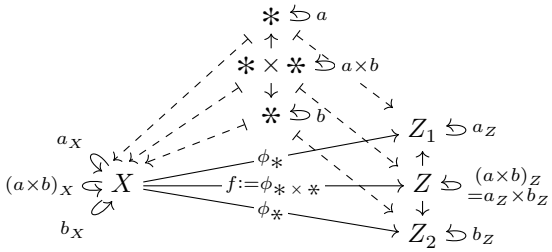
Instead, let us consider a definition based on the *product in the domain category* **S**, which could be more flexible:

Disentanglement 4 (A natural transformation at a product)**.** Let **S** be a category with binary products, **C** be a category, and $F_X, F_Z : \mathbf{S} \to \mathbf{C}$ be functors. A disentangled encoding process is a *component* of a natural transformation $\phi : F_X \Rightarrow F_Z$ at a *product*.

Additionally, if the codomain category **C** also has products, we can require that

Disentanglement 4.1. F_Z is *product-preserving*.

In other words, F_Z should be a *cartesian (monoidal) functor*, so products and projections in **S** are mapped to products and projections in **C**. An example is shown below:



(9)

We can see that two **S**-objects $\ast$ and $\circledast$ have a product $\ast \times \circledast$. F_Z preserves products so $(a \times b)_Z = a_Z \times b_Z$. A disentangled encoding process $f := \phi_{\ast\times\circledast}$ is a component of a natural transformation ϕ at a product $\ast \times \circledast$. Note that X is not necessarily a product but its endomorphisms can have a product structure (Higgins et al., 2018).

Next, let us check what the counterpart of *modularity* is in the context of natural transformations. What we will do here is essentially the same as what we showed in Section 3.2. Again, it is natural to consider a code generating process $\mu : F_Y \Rightarrow F_Z$ in $[\mathbf{S}, \mathbf{C}]$, and we have a counterpart of Assumption 1 as follows:

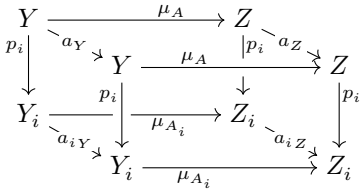
Assumption 4. F_Y is product-preserving.

Then, we can simply say that a modular encoder μ is a *natural transformation between product-preserving functors*. Even more, we can prove the following property:

Proposition 5. $\forall \ast, \circledast \in \mathbf{S}.\ \mu_{\ast\times\circledast} = \mu_\ast \times \mu_\circledast$.

The reader should compare **D**. 4.1, Assumption 4, and Proposition 5 with **D**. 1.1.

The following commutative diagram encompasses all the requirements (cf. Proposition 1):



(10)

The three axes correspond to (i) product, (ii) endomorphism, and (iii) natural transformation, respectively.

Up to this point, our definition includes the one proposed by Higgins et al. (2018) as a special case. The reader may have noticed that there is only a counterpart of modularity **D**. 1.1 but not explicitness **D**. 1.4. Without the requirement, we may encounter the same failure case:

Example (Constant)**.** The *constant functor* $\Delta 1$ satisfies **D**. 4.1 with a natural transformation $e_Y : Y \to 1$.

To patch this, one way is to require that

Disentanglement 4.2. F_Z is *faithful*.

This means that F_Z is injective on morphisms for each pair of **S**-objects. We need to rule out unfaithful models of a scheme lest we end up with uninformative representations. This requirement also tells us some basic properties the codes Z should have such as the minimal size or dimension, depending on the choice of the scheme **S**.

On the other hand, the exact counterpart of explicitness **D**. 1.4 is as follows:

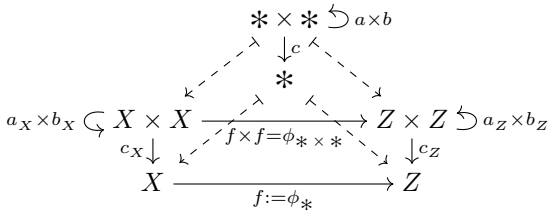
Disentanglement 4.3. μ is a split monomorphism.

D. 4.3 is a stronger notion when F_Y is also faithful:

Theorem 6. ***D***. 4.3 $\to$ ***D***. 4.2.

As a final note, we point out that **D**. 4 is more flexible because it is not limited to endomorphisms:

Example (Binary operation)**.** Let $\ast$ be an **S**-object (which itself can be a product) and $c : \ast \times \ast \to \ast$ an **S**-morphism. The following diagram describes how binary operations can exhibit disentanglement:



(11)

Regarding $c \circ (a \times b)$, the functoriality and naturality lead to the following requirement:

$$f(c_X(a_X(x_1), b_X(x_2))) = c_Z(a_Z(f(x_1)), b_Z(f(x_2))).$$

This formulation is particularly useful when dealing with multiple instances or heterogeneous inputs (Gatys et al.,

2016; Liu et al., 2018). Further investigation is left for future work.

In summary, we showed that seemingly distinct approaches to disentanglement (Ridgeway & Mozer, 2018; Higgins et al., 2018) can be described by the same abstract language, and their underlying mechanisms (e.g., modularity and product-preserving action) are essentially the same. The core is the *cartesian product* of sets, functions, algebras, actions, objects, morphisms, categories, and functors.

5. Sets and Relations

The Cartesian product of sets is not cartesian in **Rel**.

In this section, we present an example of (non-cartesian) monoidal products using **Rel**, the category of sets and relations (Patterson, 2017).

We may want to work with relations instead of functions if we need to consider (i) unannotated factors, (ii) multiple observations for the same factor, or (iii) only a subset of all combinations of factors. Besides, **Rel** serves as a bridge between functions and probabilities, which will be discussed in the next section.

To characterize **Rel**, it is convenient to consider it as the *Kleisli category* of the *powerset monad* P in **Set**:

$$\mathbf{Rel} := \mathbf{Set}_P. \tag{12}$$

The powerset monad P sends a set A to its powerset PA and a function $f : A \to B$ to a set function $Pf : PA \to PB$.[3] Its Kleisli category **Rel** has relations $A \rightsquigarrow B$ as the *Kleisli morphisms*, which are precisely set-valued functions $A \to PB$. The composition is the *Kleisli composition* $g \Leftarrow f$,[4] given by the *relation composition*.

Relations arise naturally in practice. For example, if we have a *labeling process* $l : X \to Y$, which is a function in **Set**, a data generating process $g : Y \rightsquigarrow X := l^*$ can be defined as its *inverse image*, which is not a function anymore but a relation in **Rel**. Then, g now can map a factor to multiple observations or the empty set.

5.1. Monoidal Product of Relations

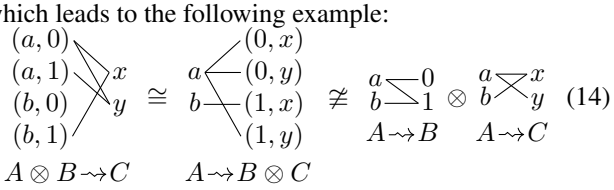
Next, let us examine the product structures in **Rel**. We point out the following three important facts about **Rel**:

- **Rel** is cartesian and cocartesian, with both the product and coproduct given by the *disjoint union* of sets $A \oplus B$.
- **Rel** is *monoidal closed*, with both the monoidal product and *internal hom* given by the *Cartesian product* of sets $A \otimes B$ and the monoidal unit given by the singleton $\{*\}$.
- **Rel** is *pointed*, with the *zero object* (an object that is both initial and terminal) given by the empty set $\varnothing$.

That is, in **Rel**, the Cartesian product of sets is *monoidal*, but confusingly, *not cartesian*. So a relation $A \rightsquigarrow B \otimes C$ to a Cartesian product of two sets is more than just a pair of relations $A \rightsquigarrow B$ and $A \rightsquigarrow C$. On the other hand, the monoidal product/internal hom $\otimes$ gives us an isomorphism between hom-sets:

$$\mathrm{Hom}(A \otimes B, C) \cong \mathrm{Hom}(A, B \otimes C), \tag{13}$$

which leads to the following example:



(14)

Rel is an example of how the cartesian product $\oplus$ is not an appropriate concept for disentanglement, while a suitable one $\otimes$ only has a monoidal structure. The monoidal unit $\{*\}$ is different from the terminal object $\varnothing$, so **Rel** is not even semicartesian. Although we still can define the "*duplicating*" and "*deleting*" operations (Patterson, 2017, Section 3.3), they do not behave as nicely as those diagonal and terminal morphisms in **Set** because of their non-naturality.

Then, how can we characterize disentanglement? At least, we still have a counterpart of disentanglement **D**. 1:

Disentanglement 5 (A morphism to a monoidal product)**.** In a symmetric monoidal category **C**, a disentangled encoding process is a morphism $f : X \to Z$ to a *monoidal product* $Z := \bigotimes_{i=1}^{N} Z_i$.

Further, we can extend the definition of modularity **D**. 1.1:

Disentanglement 5.1. $m = \bigotimes_{i=1}^{N} (m_{i,i} : Y_i \to Z_i)$.

So, **D**. 1 and **D**. 1.1 are special cases of **D**. 5 and **D**. 5.1 for a cartesian category. However, without projections, **D**. 5.1 is more difficult to verify than **D**. 1.1.

Then, how can we resolve this? One way is to restrict our attention to *right-unique relations*, i.e., *partial functions*, so duplication behaves nicely (Eq. (2)), but it means that there is at most one observation for each factor. We can also focus on *left-total relations*, i.e., *multivalued functions*, so deletion behaves nicely (Eq. (3)), but we need to assume that there is at least one observation for each factor (Fritz, 2020, Example 2.6). If we want both, then we will end up with **Set** — a cartesian subcategory of **Rel**. Despite its many good properties, **Set** might be too restrictive if we want to incorporate uncertainty in disentanglement. Later we will see that a *semicartesian category with (not necessarily natural) diagonals* might be a balanced choice, which provides a rich collection of operations to characterize disentanglement.

[3] The map on morphisms is the polymorphic function `fmap :: Functor f => (a -> b) -> f a -> f b` in Haskell.

[4] This is the "*left fish*" operator in Haskell: `(<=<) :: Monad m => (b -> m c) -> (a -> m b) -> a -> m c`.

5.2. Functor Category, Revisited

Before moving on to the next section, "*can we change from* **Rel** *to* [**S**, **Rel**]*?*" we have to ask. First, the fact that [**S**, **C**] has a pointwise monoidal structure derived from **C** tells us that **D**. 2 generalizes to the *functor monoidal product* straightforwardly. Second, **D**. 4.1 is a special case of the following requirement for a cartesian category:

Disentanglement 4.1′. F_Z is a *monoidal functor*.

Higgins et al. (2018) mainly worked with the direct sum $\oplus$ (direct product $\times$) of vector spaces and briefly mentioned the tensor product $\otimes$. We remind the reader that their decisive difference is between the cartesian and monoidal products.

6. Measurable Spaces and Stochastic Maps

We can copy/delete in a Markov category like **Stoch**.

Besides the algebraic approach (Higgins et al., 2018), the probabilistic, statistical, and information-theoretic methods (Higgins et al., 2017; Chen et al., 2018; Kumar et al., 2018; Suter et al., 2019; Do & Tran, 2020) are perhaps the most popular tools for disentangled representation learning. In this section, we outline the essential operations required for characterizing disentanglement of stochastic maps.

The structure is similar to that of **Rel**: the category **Stoch** of measurable spaces and stochastic maps (Markov kernels) is the Kleisli category of the *Giry monad* P in the category **Meas** of measurable spaces and measurable functions:

$$\mathbf{Stoch} := \mathbf{Meas}_P. \tag{15}$$

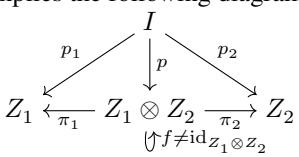
The Giry monad P sends a measurable set A to the set PA of probability measures on A and a measurable function $f : A \to B$ to its pushforward $f_* : PA \to PB$. The Kleisli morphisms are stochastic maps $p(B|A)$, and the Kleisli composition $p(C|A) = p(C|B) \Leftarrow p(B|A)$ is the *Chapman–Kolmogorov equation* (Giry, 1982).

6.1. Joint Distribution and Conditional Independence

Next, let us start by highlighting the impossibility result in Locatello et al. (2019), which is essentially about the product structures of **Stoch**. It can be succinctly restated in the categorical language as

Theorem 7 (Locatello et al. (2019, Theorem 1)). **Stoch** *is not cartesian.*

This theorem implies the following diagram (cf. Eq. (1)):



(16)

It means that a joint distribution $p(Z_1, Z_2)$ is not uniquely specified by its marginals $p_1(Z_1)$ and $p_2(Z_2)$. Locatello et al. (2019) explicitly constructed a family of bijections $f : Z \to Z$ using the inverse transform sampling technique.

Note the projection morphisms π_1 and π_2 used in Eq. (16), which are available because **Stoch** is a Markov category (Fritz, 2020). A Markov category, roughly speaking, is a category in which every object A is equipped with a "*copy*" $\mathrm{copy}_A : A \to A \otimes A$ (not necessarily natural in A) and a "*delete*" $\mathrm{del}_A : A \to I$ (natural in A) morphism satisfying some coherence conditions. Therefore, all morphisms are *deletable* but only some are *copyable*, which allows for a sufficiently expressive category with enough operations to characterize disentanglement:

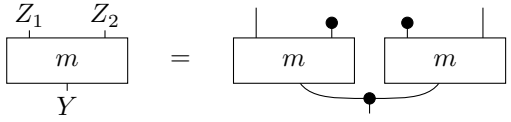
Disentanglement 6 (A Markov kernel to a joint)**.** In a Markov category **C**, a disentangled encoding process is a *Markov kernel* $f : X \to Z$ to a *joint* $Z := \bigotimes_{i=1}^{N} Z_i$.

We point out that the *conditional independence* $A \perp B \parallel C$ of a Markov kernel $p(A, B|C)$ (Fritz, 2020, Definition 12.12) can be used to derive a prerequisite for the modularity of an encoder $m : Y \to Z$:

Disentanglement 6.1. $\forall i \in [1..N].\ Z_i \perp Z_{\setminus i} \parallel Y$.

Let $m_i : Y \to Z_i := \mathrm{del}_{Z_{\setminus i}} \circ m$ be the *marginalization* of m over $Z_{\setminus i}$ and $\mathrm{copy}_A^N : A \to A^{\otimes N}$ a "*multiple copy*" morphism. We can prove that **D**. 6.1 is equivalent to the following equational identity (cf. Eq. (1)):

Disentanglement 6.2. $m = (\bigotimes_{i=1}^{N} m_i) \circ \mathrm{copy}_Y^N$.

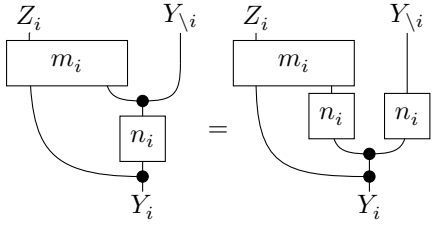


Theorem 8. ***D***. 6.1 $\leftrightarrow$ ***D***. 6.2.

We call an encoder satisfying **D**. 6.2 *projectable*. This is a more fine-grained condition than the *total correlation* used in Chen et al. (2018) because it is conditioned on the factors.

With this precondition, we can finally define the modularity of a stochastic encoder:

Disentanglement 6.3. $\forall i \in [1..N].\ \forall n_i : Y_i \to Y_{\setminus i}.\ Z_i \perp Y_{\setminus i} \parallel Y_i$.



The reader may have noticed that this means that any $n_i : Y_i \to Y_{\setminus i}$ behaves like a *deterministic morphism* (Fritz, 2020, Definition 10.1) when composed with $m_i : Y \to Z_i$.

Why do we need this? It is because, not like $\mathbf{Rel}$, where $A \otimes B \rightsquigarrow C$ is the same thing as $A \rightsquigarrow B \otimes C$ (Eq. (13)), in $\mathbf{Stoch}$, $\mathrm{Hom}(A, B \otimes C)$ is larger than $\mathrm{Hom}(A \otimes B, C)$. We need a "*probe*" in $\mathrm{Hom}(A, B)$ to characterize how C depends on B, and $n_i : Y_i \to Y_{\setminus i}$ serves as this probe.

Based on this, we revealed a common loophole in existing statistical approaches: if we use the mutual information between factors and codes to characterize disentanglement, the distribution of factors is assumed to be fixed (Chen et al., 2018; Li et al., 2020; Tokui & Sato, 2022). However, the training and test distributions could be different (Träuble et al., 2021), and the existing definitions may be too coarse-grained and insufficient in such a situation.

6.2. Structured Markov Kernels

An important fact is that *the category of functors to the subcategory of deterministic morphisms is again a Markov category* (Fritz, 2020, Section 7), so we can deal with "*structured*" Markov kernels. We end our discussion with an example based on this fact, without any category theory jargon, to show what we can get from our formulation.

Example ($[\mathbb{N}, \mathbf{Set}_N]_{\mathrm{det}}$)**.** A robot processing video feed of multiple objects should be able to (i) identify objects, (ii) understand that objects continue to exist even if they are occluded (*object permanence*), and (iii) track the movement of hidden objects (*invisible displacement*). All these abilities should not be affected by the shape or color of the objects.

With category theory, we can formulate such a complex task with ease because the components are *compositional*. See Appendix B for a detailed explanation.

7. Limitations

As an initial step toward categorical characterization of disentanglement, this work only focused on the definitions. Many other important aspects of disentanglement were excluded, such as metrics, supervision, learnability, models, methods, and experimental validation.

With a clear understanding of the definitions in place, the immediate next step would be to find a systematic way to *enrich a definition to a metric*. We hypothesize that a good direction includes the following three steps:

- equality $\rightsquigarrow$ metric
- universal quantification $\rightsquigarrow$ aggregation
- existential quantification $\rightsquigarrow$ approximation

With a good metric, we can quantify how good a model is, even if it does not strictly satisfy a disentanglement definition. For example, from Theorem 4, we know that a modular and explicit encoder must have a modular decoder. Given some modularity and explicitness metrics, we may want to know *how much* the modularity and explicitness of an encoder imply the modularity of its decoder.

Other potential future directions include the studies of partial combination of factors (Section 5) and unknown projection (Assumption 3). The relation between **D**. 2, **D**. 3, and **D**. 4 deserves further investigation. How to formulate disentanglement in more complex learning problems, such as reinforcement learning, is also an interesting direction. While we have obtained more results for cartesian categories due to their favorable properties, further theoretical analyses on the monoidal category case would be useful.

8. Conclusion

In this work, we presented a meta-analysis of several definitions of disentanglement (Cohen & Welling, 2014; 2015; Ridgeway & Mozer, 2018; Eastwood & Williams, 2018; Higgins et al., 2018; Chen et al., 2018) using *category theory* as a unifying language. We revealed that some seemingly distinct formulations are just different manifestations of the same structures, the *cartesian products and monoidal products*, in different categories. We argued that the modularity (*product morphism*) and explicitness (*split monomorphism*) should be the defining properties of disentanglement and introduced tools to analyze these properties in various settings, including equivariant maps (*functor category*) and stochastic maps (*Markov category*). We also reinterpreted some existing results (Locatello et al., 2019) and provided support to some arguments based on empirical evidence (Ridgeway & Mozer, 2018; Träuble et al., 2021). We hope our findings can help researchers choose the most appropriate definition of disentanglement for their specific task and consequently discover better metrics, models, methods, and algorithms for disentangled representation learning.

Acknowledgements

We would like to thank Tobias Fritz for answering our questions about Markov categories. We thank Wei Wang and Johannes Ackermann for their valuable feedback on the draft. We also thank the anonymous reviewers for their useful comments and constructive suggestions. Finally, we would like to express our gratitude to all contributors to nLab, MathOverflow, and StackExchange for creating a sharing community.

YZ was supported by JSPS KAKENHI Grant Number 22J12703. MS was supported by JST CREST Grant Number JPMJCR18A2.

References

Adámek, J., Herrlich, H., and Strecker, G. E. *Abstract and Concrete Categories: The Joy of Cats*. John Wiley and Sons, 1990. URL http://www.tac.mta.ca/tac/reprints/articles/17/tr17abs.html. A.5

Awodey, S. *Category theory*. Oxford University Press, 2006. URL https://doi.org/10.1093/acprof:oso/9780198568612.001.0001. 1, 2, A.1

Baez, J. Applied category theory 2018 | the n-category café, 2017. URL https://golem.ph.utexas.edu/category/2017/09/applied_category_theory_1.html. 1

Baez, J. C., Fritz, T., and Leinster, T. A characterization of entropy in terms of information loss. *Entropy*, 13(11): 1945–1957, 2011. URL https://doi.org/10.3390/e13111945. https://arxiv.org/abs/1106.1791. A.5

Bengio, Y., Courville, A., and Vincent, P. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828, 2013. URL https://doi.org/10.1109/TPAMI.2013.50. https://arxiv.org/abs/1206.5538. 1

Borceux, F. *Handbook of categorical algebra: volume 1, Basic category theory*, volume 1. Cambridge University Press, 1994. URL https://doi.org/10.1017/CBO9780511525858. 1

Bradley, T.-D. What is applied category theory? *arXiv preprint arXiv:1809.05923*, 2018. URL https://arxiv.org/abs/1809.05923. 1

Carbonneau, M.-A., Zaidi, J., Boilard, J., and Gagnon, G. Measuring disentanglement: A review of metrics. *IEEE Transactions on Neural Networks and Learning Systems*, 2022. URL https://doi.org/10.1109/TNNLS.2022.3218982. https://arxiv.org/abs/2012.09276. 1, 3.3

Chen, R. T., Li, X., Grosse, R. B., and Duvenaud, D. K. Isolating sources of disentanglement in variational autoencoders. In *Neural Information Processing Systems*, 2018. URL https://proceedings.neurips.cc/paper/2018/hash/1ee3dfcd8a0645a25a35977997223d22-Abstract.html. 1, 6, 6.1, 6.1, 8

Cho, K. and Jacobs, B. Disintegration and bayesian inversion via string diagrams. *Mathematical Structures in Computer Science*, 29(7):938–971, 2019. URL https://doi.org/10.1017/S0960129518000488. https://arxiv.org/abs/1709.00322. A.7

Cohen, T. and Welling, M. Learning the irreducible representations of commutative lie groups. In *International Conference on Machine Learning*, 2014. URL https://proceedings.mlr.press/v32/cohen14.html. 1, 8, A.4, A.4

Cohen, T. and Welling, M. Transformation properties of learned visual representations. In *International Conference on Learning Representations*, 2015. URL http://arxiv.org/abs/1412.7659. 1, 8, A.4

de Haan, P., Cohen, T., and Welling, M. Natural graph networks. *Neural Information Processing Systems*, 33: 3636–3646, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/2517756c5a9be6ac007fe9bb7fb92611-Abstract.html. 1

Dittadi, A., Träuble, F., Locatello, F., Wuthrich, M., Agrawal, V., Winther, O., Bauer, S., and Schölkopf, B. On the transfer of disentangled representations in realistic settings. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=8VXvj1QNRl1. 1

Do, K. and Tran, T. Theory and evaluation metrics for learning disentangled representations. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=HJgK0h4Ywr. 6

Duan, S., Matthey, L., Saraiva, A., Watters, N., Burgess, C., Lerchner, A., and Higgins, I. Unsupervised model selection for variational disentangled representation learning. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=SyxL2TNtvr. 3.3

Dudzik, A. J. and Veličković, P. Graph neural networks are dynamic programmers. In *Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=wu1Za9dY1GY. 1

Eastwood, C. and Williams, C. K. A framework for the quantitative evaluation of disentangled representations. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=By-7dz-AZ. 1, 3.2, 3.3, 3.3, 8

Fong, B. and Spivak, D. I. *An invitation to applied category theory: seven sketches in compositionality*. Cambridge University Press, 2019. URL https://doi.org/10.1017/9781108668804. https://arxiv.org/abs/1803.05316. 1

Franz, U. What is stochastic independence? In *Non-commutativity, infinite-dimensionality and probability at*

the crossroads, pp. 254–274. World Scientific, 2002. URL https://doi.org/10.1142/9789812705242_0008. https://arxiv.org/abs/math/0206017. 2.2

Fritz, T. A synthetic approach to markov kernels, conditional independence and theorems on sufficient statistics. *Mathematics*, 370:107239, 2020. URL https://doi.org/10.1016/j.aim.2020.107239. https://arxiv.org/abs/1908.07021. 1, 2.2, 5.1, 6.1, 6.1, 6.1, 6.2, A.6, 4, B, C

Gatys, L. A., Ecker, A. S., and Bethge, M. Image style transfer using convolutional neural networks. In *Computer Vision and Pattern Recognition*, 2016. URL https://doi.org/10.1109/CVPR.2016.265. 4.2

Gavranović, B. Compositional deep learning. Master's thesis, University of Zagreb, 2019. URL https://arxiv.org/abs/1907.08292. 1

Giry, M. A categorical approach to probability theory. *Categorical Aspects of Topology and Analysis*, pp. 68–85, 1982. URL https://doi.org/10.1007/BFb0092872. 6

Higgins, I., Matthey, L., Pal, A., Burgess, C., Glorot, X., Botvinick, M., Mohamed, S., and Lerchner, A. beta-VAE: Learning basic visual concepts with a constrained variational framework. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=Sy2fzU9gl. 1, 6

Higgins, I., Amos, D., Pfau, D., Racaniere, S., Matthey, L., Rezende, D., and Lerchner, A. Towards a definition of disentangled representations. *arXiv preprint arXiv:1812.02230*, 2018. URL https://arxiv.org/abs/1812.02230. 1, 4, 4.1, 4.2, 4.2, 4.2, 5.2, 6, 8, A.4, A.4

Kumar, A., Sattigeri, P., and Balakrishnan, A. Variational inference of disentangled latent concepts from unlabeled observations. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=H1kG7GZAW. 6

Leinster, T. *Basic category theory*, volume 143. Cambridge University Press, 2014. URL https://doi.org/10.1017/CBO9781107360068. https://arxiv.org/abs/1612.09375. 1

Leinster, T. Monoidal categories with projections | the n-category café, 2016. URL https://golem.ph.utexas.edu/category/2016/08/monoidal_categories_with_proje.html. 2.2

Li, Z., Murkute, J. V., Gyawali, P. K., and Wang, L. Progressive learning and disentanglement of hierarchical representations. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=SJxpsxrYPS. 6.1

Liu, T.-Y. *Learning to Rank for Information Retrieval*. Springer, 2011. URL https://doi.org/10.1007/978-3-642-14267-3. 4

Liu, X., Van De Weijer, J., and Bagdanov, A. D. Leveraging unlabeled data for crowd counting by learning to rank. In *Computer Vision and Pattern Recognition*, 2018. URL https://doi.org/10.1109/CVPR.2018.00799. 4.2

Locatello, F., Bauer, S., Lucic, M., Raetsch, G., Gelly, S., Schölkopf, B., and Bachem, O. Challenging common assumptions in the unsupervised learning of disentangled representations. In *International Conference on Machine Learning*, 2019. URL https://proceedings.mlr.press/v97/locatello19a.html. 1, 6.1, 7, 6.1, 8

Montero, M. L., Ludwig, C. J., Costa, R. P., Malhotra, G., and Bowers, J. The role of disentanglement in generalisation. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=qbH974jKUVy. 1

Patterson, E. Knowledge representation in bicategories of relations. *arXiv preprint arXiv:1706.00526*, 2017. URL https://arxiv.org/abs/1706.00526. 5, 5.1

Piedeleu, R. and Zanasi, F. An introduction to string diagrams for computer scientists. *arXiv preprint arXiv:2305.08768*, 2023. URL https://arxiv.org/abs/2305.08768. A.1

Ridgeway, K. and Mozer, M. C. Learning deep disentangled embeddings with the f-statistic loss. In *Neural Information Processing Systems*, 2018. URL https://proceedings.neurips.cc/paper/2018/hash/2b24d495052a8ce66358eb576b8912c8-Abstract.html. 1, 3.2, 3.2, 3.3, 3.3, 3.3, 4.2, 8

Selinger, P. A survey of graphical languages for monoidal categories. In *New structures for physics*, pp. 289–355. Springer, 2010. URL https://doi.org/10.1007/978-3-642-12821-9_4. https://arxiv.org/abs/0908.3347. 2, A.1

Sepliarskaia, A., Kiseleva, J., and de Rijke, M. Evaluating disentangled representations. *arXiv preprint arXiv:1910.05587*, 2019. URL https://arxiv.org/abs/1910.05587. 1

Shiebler, D., Gavranović, B., and Wilson, P. Category theory in machine learning. *arXiv preprint arXiv:2106.07032*, 2021. URL https://arxiv.org/abs/2106.07032. 1

Shu, R., Chen, Y., Kumar, A., Ermon, S., and Poole, B. Weakly supervised disentanglement with guarantees. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=HJgSwyBKvr. 3.3

Suter, R., Miladinovic, D., Schölkopf, B., and Bauer, S. Robustly disentangled causal mechanisms: Validating deep representations for interventional robustness. In *International Conference on Machine Learning*, 2019. URL http://proceedings.mlr.press/v97/suter19a.html. 6

Tokui, S. and Sato, I. Disentanglement analysis with partial information decomposition. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=pETy-HVvGtt. 6.1

Träuble, F., Creager, E., Kilbertus, N., Locatello, F., Dittadi, A., Goyal, A., Schölkopf, B., and Bauer, S. On disentangled representations learned from correlated data. In *International Conference on Machine Learning*, 2021. URL https://proceedings.mlr.press/v139/trauble21a.html. 6.1, 8

Wang, T., Yue, Z., Huang, J., Sun, Q., and Zhang, H. Self-supervised learning disentangled group representation as feature. In *Neural Information Processing Systems*, 2021. URL https://openreview.net/forum?id=RQfcckT1M_4. 3.3

A. Additional Remarks

A.1. Diagram

We frequently use commutative diagrams (Awodey, 2006) and string diagrams (Selinger, 2010; Piedeleu & Zanasi, 2023) as graphical calculus.

In a commutative diagram for a category, nodes are objects, arrows are morphisms, and paths are compositions of morphisms. Any morphisms with the same domains and codomains are equal, i.e., any two paths starting from A and ending with B result in the same morphism.

In a string diagram for a symmetric monoidal category, rectangles are morphisms (from bottom to top), lines without rectangles are identity morphisms. Juxtaposition of two morphisms means the monoidal product, and cross means the braiding. Domains and codomains of morphisms are often omitted.

A.2. Compactness

Note that there could be two interpretations of compactness. A non-compact encoder can mean:

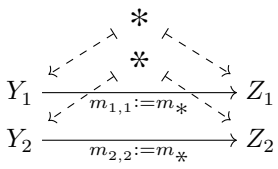
(a) We can recover Y_j from Z_i; or
(b) Z_i itself is a product $Z_{i1} \times Z_{i2}$.

We argue that (a) is what we want to avoid, while (b) is more or less harmless. For example, we can decompose $\mathbb{R}^{10}$ into $\mathbb{R}^2 \times \mathbb{R}^3 \times \mathbb{R}^5$, where each component again can be decomposed into smaller parts. However, sometimes this is beneficial: while embedding a cycle into a vector space, $\mathbb{R}^2$ may be a better choice than $\mathbb{R}$ because the embedding can be continuous. In this work, we do not pay much attention to whether each code Z_i is "minimal".

A.3. Functorial Semantics

Can we formulate modularity in terms of functors and natural transformations? The answer is yes, because the product, as a limit, can be defined via functors in the first place. Here, we only give an alternative definition of **D**. 1.1:

Disentanglement 1.1′. m is a natural transformation between functors from a *discrete category*.



(17)

It shows that a modular encoder m is merely a collection of component morphisms $m_{i,i} : Y_i \to Z_i$. Nothing more, nothing less.

A.4. Commutativity and Irreducibility

Cohen & Welling (2014) in their seminal paper used the *irreducible* representations of compact *commutative* Lie groups to define and study disentangled representations, while Higgins et al. (2018) used the *direct product* of groups. Here, we briefly remark on the product, commutativity, and irreducibility.

First, let us keep it simple and only consider *unital magma* — a set with a unital binary operation. If we have two unital magmas $(M, \circ_M, e_M)$ and $(N, \circ_N, e_N)$, we can define their product, denoted by $P = M \times N$, as the Cartesian product of their underlying sets equipped with a binary operation $\circ_P : (M \times N) \times (M \times N) \to (M \times N)$ given by

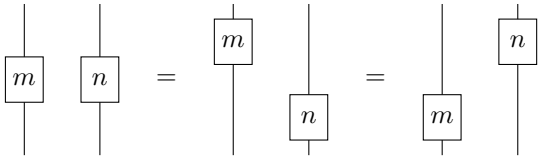
$$(m_1, n_1) \circ_P (m_2, n_2) := (m_1 \circ_M m_2, n_1 \circ_N n_2), \quad (18)$$

whose unit is $e_P := (e_M, e_N)$. We can see that the product is also a unital magma.

Then, we can find that every element (m, n) in the product P can be decomposed in two ways:

$$\begin{aligned}&(m, n)\\ =&(m \circ_M e_M, e_N \circ_N n) = (m, e_N) \circ_P (e_M, n) \quad (19)\\ =&(e_M \circ_M m, n \circ_N e_N) = (e_M, n) \circ_P (m, e_N),\end{aligned}$$

which can be expressed in string diagrams:



(20)

We can identify (m, e_N) as m and (e_M, n) as n because of the unit magma isomorphisms:

$$(M, \circ_M, e_M) \cong (M \times \{e_N\}, \circ_P|_{M \times \{e_N\}}, e_P), \quad (21)$$

$$(N, \circ_N, e_N) \cong (\{e_M\} \times N, \circ_P|_{\{e_M\} \times N}, e_P). \quad (22)$$

From this perspective, as long as we can define a serial combination $\circ$ and its unit e for each component, the product operation $\times$ allows us to combine elements from different components in parallel commutatively. We can deal with one component at a time, and the order of the components does not matter. However, note that the serial combination within a component may not be commutative, such as the 3D rotations $\mathrm{SO}(3)$ (Cohen & Welling, 2015; Higgins et al., 2018).

This property may inspire us to "*discover*" disentangled components from observational data using commutativity: we can find components such that elements from the same component are closed under composition, and elements from different components are commutative.

Such a decomposition may not be unique. For example, consider $\mathbb{R}^2$ with addition $+$ (as a unital magma, a monoid, or a group). $A = \{(a, 0) \mid a \in \mathbb{R}\}$, $B = \{(0, b) \mid b \in \mathbb{R}\}$, and $C = \{(c, c) \mid c \in \mathbb{R}\}$ are all subalgebras of $\mathbb{R}^2$, and both $A \times B$ and $A \times C$ are isomorphic to $\mathbb{R}^2$ via the addition.

Learning the (potentially product) algebraic structure from data and determining an appropriate decomposition based on commutativity is an interesting research direction.

Besides, it is worth noting that Cohen & Welling (2014) identified a connection between irreducible representations and disentanglement, which is not covered in this work. Furthermore, Cohen & Welling (2015) made an insightful observation that irreducibility is also linked to the statistical dependency structure of the representation. Using tools such as functor categories and Markov categories, we may obtain more fruitful results on the connection between algebraic and statistical properties of disentanglement.

A.5. Probability

Note that **Meas** has finite products: $(A, \Sigma_A) \times (B, \Sigma_B) := (A \times B, \Sigma_A \times \Sigma_B)$, where $\Sigma_A \times \Sigma_B$ is the coarsest σ-algebra such that two projections are measurable.

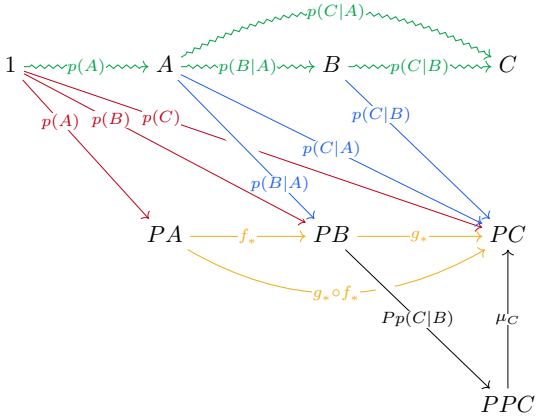
A useful construction is the category of probability measures and measure-preserving functions, which can be defined based on the concept of the *comma category*:

$$\mathbf{Prob} := \mathbf{1} \hookrightarrow \mathbf{Stoch} \downarrow \mathbf{Meas} \to \mathbf{Stoch}. \tag{23}$$

Concretely, $\mathbf{1} \hookrightarrow \mathbf{Stoch}$ is the inclusion functor, and the functor $\mathbf{Meas} \to \mathbf{Stoch}$ sends a measurable set A to itself and a measurable function f to its pushforward f_*.

Prob is a category whose objects are (isomorphic to) probability measures $(A, 1 \xrightarrow{p_A} PA)$, and morphisms $p_A \to p_B$ are measure-preserving functions $f : A \to B$ such that $p_B = f_* p_A$. This category will be important when characterizing the metrics based on entropy and mutual information (Baez et al., 2011).

Meas, **Stoch**, and **Prob** can be illustrated as follows:



(24)

All arrows are morphisms in **Meas**; red arrows are *objects* in **Prob**; yellow arrows are morphisms in **Prob**; green squiggly arrows represent morphisms in **Stoch**, which are the same as red or blue arrows.

The commutativity of red and yellow arrows indicates the composition of measure-preserving functions in **Prob**; the commutativity of blue and black arrows indicates the Kleisli composition of stochastic maps in **Stoch**.

As a side note, we can also use this construction to define the category of relations and relation-preserving functions (Adámek et al., 1990, Secion 3.3):

$$\mathbf{1} \hookrightarrow \mathbf{Rel} \downarrow \mathbf{Set} \to \mathbf{Rel}. \tag{25}$$

A.6. Markov Category

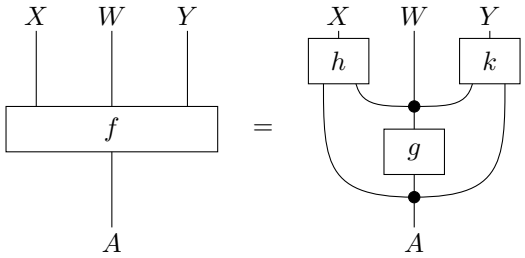
A Markov category (Fritz, 2020) is a symmetric monoidal category in which every object A is equipped with a *commutative comonoid* structure given by a *comultiplication* $\mathrm{copy}_A : A \to A \otimes A$ and a *counit* $\mathrm{del}_A : A \to I$, depicted in string diagrams as

copy_A =  del_A =  (26)

and satisfying some compatibility conditions.

A.7. Conditional Independence

Definition 4 (Conditional independence (Fritz, 2020, Definition 12.16))**.** A morphism $f : A \to X \otimes W \otimes Y$ displays the *conditional independence* $X \perp Y \mid W \parallel A$ if there are morphisms $g : A \to W$, $h : A \otimes W \to X$ and $k : W \otimes A \to Y$ such that

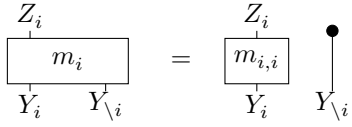


(27)

Two special cases are when $A = I$ we have $X \perp Y \mid W$ and when $W = I$ we have $X \perp Y \parallel A$.

Another way to define the modularity of a stochastic encoder is as follows, which relies on the existence of some other morphisms (cf. **D**. 1.2):

Disentanglement 6.4. $\forall i \in [1..N].\ m_i = m_{i,i} \otimes \mathrm{del}_{Y_{\setminus i}}$.



This condition was also studied in Cho & Jacobs (2019, Proposition 6.9). We can see that it is stronger than **D**. 6.3:

Theorem 9. ***D***. 6.4 $\to$ ***D***. 6.3.

However, it is not yet clear if they are equivalent.

B. Example

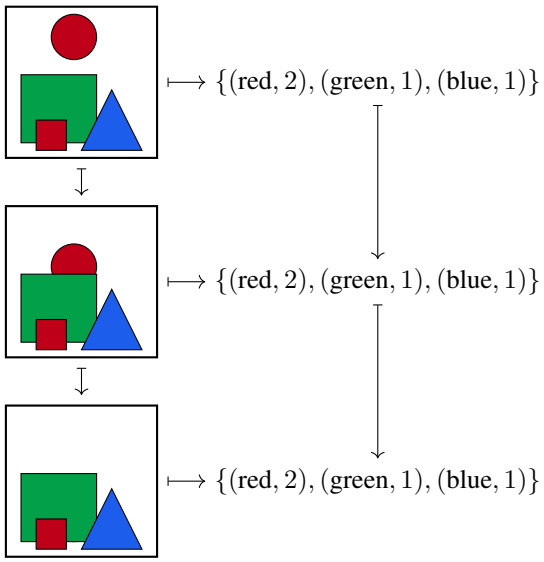
Let us start from the category **Set**. Consider the nonempty multiset monad N in **Set**, which sends a set A to $\mathbb{N}^A \setminus \varnothing$. For example, the set $\{a, b\}$ is sent to $\{\{(a,1)\}, \{(a,2)\}, \ldots, \{(b,1)\}, \ldots, \{(a,1),(b,1)\}, \ldots\}$

The Kleisli category $\mathbf{Set}_N$ of this monad consists of sets and multiset functions. A multiset function $f : A \rightsquigarrow B$ outputs how many ways to get a target $b \in B$ from a source $a \in A$. The composition of multiset functions is defined by the multiplication and sum of natural numbers. This category is a Markov category.

A Markov category $\mathbf{C}$ has a cartesian subcategory $\mathbf{C}_{\text{det}}$ of deterministic morphisms. Given a small category $\mathbf{S}$, the subcategory $[\mathbf{S}, \mathbf{C}]_{\text{det}}$ of the functor category $[\mathbf{S}, \mathbf{C}]$, which consists of functors of the form $\mathbf{S} \to \mathbf{C}_{\text{det}} \hookrightarrow \mathbf{C}$, is again a Markov category (Fritz, 2020, Section 7, notation slightly modified). The category $[\mathbf{S}, \mathbf{C}]_{\text{det}}$ contains deterministic diagrams of shape $\mathbf{S}$ and stochastic maps between them that preserve the shape.

The set of natural numbers can be considered a single-object category $(*, \mathbb{N}, +, 0)$ with the numbers as morphisms and the addition as the composition. The identity morphism id_* is the number 0.

Based on these, let us consider the category $[\mathbb{N}, \mathbf{Set}_N]_{\text{det}}$. This category contains sets equipped with endofunctions indexed by natural numbers as objects and multiset functions between these sets that preserve their endofunctions as morphisms. A natural transformation to a constant functor (which maps all morphisms to the identity morphism) in this category means that no matter how the input changes with time, the count is invariant. An example is shown bellow:



If we want to characterize more complex behavior, we may simply change the source category $\mathbb{N}$ and define a proper category (possibly with a product structure) that encodes our requirements. The extension is left for future work.

C. Proofs

Proposition 1.

$$\begin{array}{ccccc} Y_1 & \xleftarrow{p_1} & Y_1 \times Y_2 & \xrightarrow{p_2} & Y_2 \\ \downarrow{\scriptstyle m_{1,1}} & & \downarrow{\scriptstyle m_{1,1} \times m_{2,2}} & & \downarrow{\scriptstyle m_{2,2}} \\ Z_1 & \xleftarrow[p_1]{} & Z_1 \times Z_2 & \xrightarrow[p_2]{} & Z_2 \end{array} \tag{28}$$

□

Proposition 2.

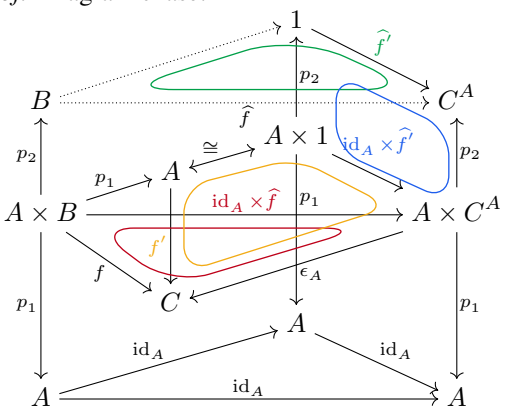
$$\begin{array}{ccc} Y_1 & \xleftarrow{p_1} & Y_1 \times 1 \\ \downarrow{\scriptstyle \text{id}_{Y_1}} & \nearrow_{\cong} & \downarrow{\scriptstyle \text{id}_{Y_1} \times y_2} \\ Y_1 & \xleftarrow[p_1]{} & Y_1 \times Y_2 \\ \downarrow{\scriptstyle m_{1,1}} & & \downarrow{\scriptstyle m} \\ Z_1 & \xleftarrow[p_1]{} & Z_1 \times Z_2 \end{array} \tag{29}$$

□

Theorem 3 can be proven using the following lemma:

Lemma 10. *Let $f : A \times B \to C$ be a morphism from a product and $\widehat{f} : B \to C^A$ its exponential transpose. Then, there exists a morphism $f' : A \to C$ such that $f = f' \circ p_1$ if and only if the exponential transpose $\widehat{f}$ is a constant morphism.*

Proof. Diagram chase:



(30)

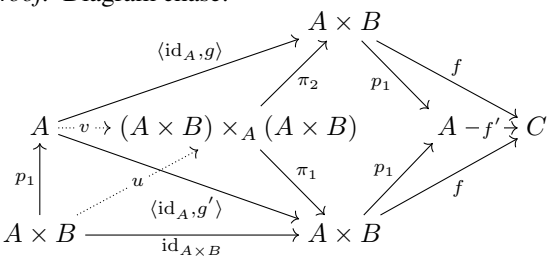
We need to use the following commutative diagrams: (i) red: the universal property of the exponential object C^A and the evaluation morphism ϵ_A; (ii) green: the constant morphism $\widehat{f}$, which factors through the terminal object 1 and defines the morphism $\widehat{f'}$; (iii) blue: the product morphism $\text{id}_A \times \widehat{f'}$; and (iv) yellow: the definition of f'.

It is easy to prove $\widehat{f} : B \to C^A$ is a constant morphism if $f = f' \circ p_1$. Suppose $\widehat{f} : B \to C^A$ is a constant morphism, so it factors through the terminal object 1. We denote the morphism by $\widehat{f'} : 1 \to C^A$. We can define $f' : A \to C$ as $\epsilon_A \circ (\mathrm{id}_A \times \widehat{f'})$. To prove $f = f' \circ p_1$, i.e., $f = \epsilon_A \circ (\mathrm{id}_A \times \widehat{f'}) \circ p_1$, we only need to show $\mathrm{id}_A \times \widehat{f} = (\mathrm{id}_A \times \widehat{f'}) \circ (\mathrm{id}_A \times e_B)$. This triangle commutes because it is simply a product of the identity morphism id_A and the constant morphism $\widehat{f}$. □

Alternatively, we can also characterize product morphisms using *pullback*. Concretely, let $Y \times_{Y_i} Y$ be the pullback of the projections $p_i : Y \to Y_i$ and $\pi_1, \pi_2 : Y \times_{Y_i} Y \to Y$ be the *pullback projections*. In the category **Set** of sets, the pullback $Y \times_{Y_i} Y = \{(y, y') \in Y \times Y \mid y_i = y'_i\}$ is the set of pairs of factors whose i-th components are the same. Then, m is a product morphism if and only if $m_i \circ \pi_1 = m_i \circ \pi_2$, i.e., $m_i(y_i, y_{\setminus i}) = m_i(y_i, y'_{\setminus i})$. This can be proven using the following lemma:

Lemma 11. *Let $f : A \times B \to C$ be a morphism from a product and $(A \times B) \times_A (A \times B)$ be the pullback of the projections $p_1 : A \times B \to A$ with two pullback projections $\pi_1, \pi_2 : (A \times B) \times_A (A \times B) \to A \times B$. Then, there exists a morphism $f' : A \to C$ such that $f = f' \circ p_1$ if and only if $f \circ \pi_1 = f \circ \pi_2$.*

Proof. Diagram chase:



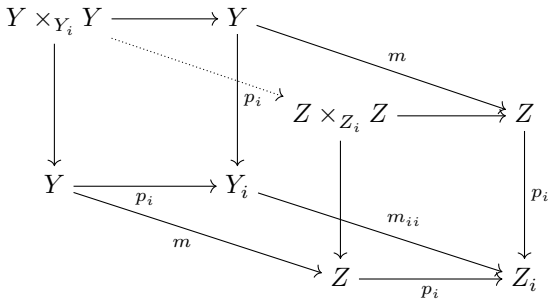
(31)

Suppose that $f = f' \circ p_1$. Because the pullback rectangle commutes, $p_1 \circ \pi_1 = p_1 \circ \pi_2$, it is easy to show that $f \circ \pi_1 = f' \circ p_1 \circ \pi_1 = f' \circ p_1 \circ \pi_2 = f \circ \pi_2$.

Now suppose that $f \circ \pi_1 = f \circ \pi_2$. We define $f' : A \to C$ as $f \circ \langle \mathrm{id}_A, g \rangle$ for an arbitrary morphism $g : A \to B$. To prove $f = f' \circ p_1$, we can consider two morphisms $\mathrm{id}_{A\times B}$ and $\langle \mathrm{id}_A, g \rangle \circ p_1$. Because they complete the commutative diagram of the pullback $(A \times B) \times_A (A \times B)$, $p_1 \circ \mathrm{id}_{A\times B} = p_1 \circ \langle \mathrm{id}_A, g \rangle \circ p_1 = p_1$, there exists a unique morphism $u : A \times B \to (A \times B) \times_A (A \times B)$ such that $\pi_1 \circ u = \mathrm{id}_{A\times B}$ and $\pi_2 \circ u = \langle \mathrm{id}_A, g \rangle \circ p_1$. We can now chase the diagram to show that $f = f \circ \mathrm{id}_{A\times B} = f \circ \pi_1 \circ u = f \circ \pi_2 \circ u = f \circ \langle \mathrm{id}_A, g \rangle \circ p_1 = f' \circ p_1$.

To prove that this construction does not depend on specific choice of $g : A \to B$, let us consider two morphisms $g, g' : A \to B$. Because $\langle \mathrm{id}_A, g \rangle$ and $\langle \mathrm{id}_A, g' \rangle$ complete the commutative diagram of the pullback, there exists a unique morphism $v : A \to (A \times B) \times_A (A \times B)$ such that $\pi_1 \circ v = \langle \mathrm{id}_A, g' \rangle$ and $\pi_2 \circ v = \langle \mathrm{id}_A, g \rangle$. Then, $f \circ \langle \mathrm{id}_A, g \rangle = f \circ \pi_2 \circ v = f \circ \pi_1 \circ v = f \circ \langle \mathrm{id}_A, g' \rangle$, which shows that $f' = f \circ \langle \mathrm{id}_A, g \rangle$ is independent of the choice of $g : A \to B$. □

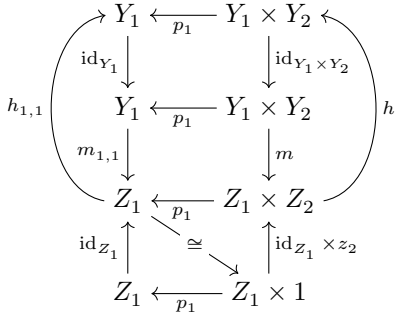
Based on this, we can obtain the following diagram:



(32)

Both Lemmas 10 and 11 show that there are alternative ways to characterize "*invariance*", without a group theoretical formulation.

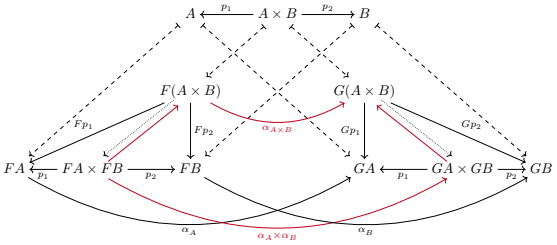
Theorem 4.



(33)

□

Proposition 5. Let $F, G : \mathbf{C} \to \mathbf{D}$ be product preserving functors.



(34)

□

Theorem 6. Let $F, G : \mathbf{C} \to \mathbf{D}$ be functors, $\alpha : F \Rightarrow G$ be a natural transformation.

$$\begin{array}{ccc} FA & \xrightarrow{\alpha_A} & GA \\ {\scriptstyle Fp,Fq}\downarrow & & \downarrow{\scriptstyle Gp,Gq} \\ FB & \xrightarrow[\alpha_B]{} & GB \end{array} \tag{35}$$

We have the following reasoning:

- F is not faithful: $\exists p \neq q.\ Fp = Fq$
- α is natural: $Fp = Fq \to Gp \circ \alpha_A = Gq \circ \alpha_A$
- α is epic: $Gp \circ \alpha_A = Gq \circ \alpha_A \to Gp = Gq$

Then,

$$F \text{ is not faithful} \wedge \alpha \text{ is epic} \to G \text{ is not faithful.} \tag{36}$$

Or equivalently,

$$\alpha \text{ is epic} \to (G \text{ is faithful} \to F \text{ is faithful}). \tag{37}$$

Similarly,

- G is not faithful: $\exists p \neq q.\ Gp = Gq$
- α is natural: $Gp = Gq \to \alpha_B \circ Fp = \alpha_B \circ Fq$
- α is monic: $\alpha_B \circ Fp = \alpha_B \circ Fq \to Fp = Fq$

Then,

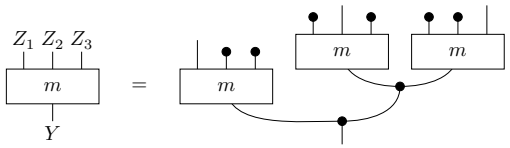
$$G \text{ is not faithful} \wedge \alpha \text{ is monic} \to F \text{ is not faithful.} \tag{38}$$

Or equivalently,

$$\alpha \text{ is monic} \to (F \text{ is faithful} \to G \text{ is faithful}). \tag{39}$$

□

Theorem 8. When $N = 2$, **D**. 6.2 is the definition of **D**. 6.1 (Fritz, 2020, Lemma 12.11). When $N > 2$, we can apply this equation recursively.



(40)

□