GAIA: CATEGORICAL FOUNDATIONS OF GENERATIVE AI*

A PREPRINT

Sridhar Mahadevan
Adobe Research and University of Massachusetts, Amherst
smahadev@adobe.com, mahadeva@umass.edu

February 16, 2024

ABSTRACT

In this paper, we explore the categorical foundations of generative AI. Specifically, we investigate a Generative AI Architecture (GAIA) that lies beyond backpropagation, the longstanding algorithmic workhorse of deep learning. Backpropagation is at its core a compositional framework for (un)supervised learning: it can be conceptualized as a sequence of modules, where each module updates its parameters based on information it receives from downstream modules, and in turn, transmits information back to upstream modules to guide their updates. GAIA is based on a fundamentally different *hierarchical model*. Modules in GAIA are organized into a simplicial complex. Each n-simplicial complex acts like a manager of a business unit: it receives updates from its superiors and transmits information back to its $n+1$ subsimplicial complexes that are its subordinates. To ensure this simplicial generative AI organization behaves coherently, GAIA builds on the mathematics of the higher-order category theory of simplicial sets and objects. Computations in GAIA, from query answering to foundation model building, are posed in terms of lifting diagrams over simplicial objects. The problem of machine learning in GAIA is modeled as "horn" extensions of simplicial sets: each sub-simplicial complex tries to update its parameters in such a way that a lifting diagram is solved. Traditional approaches used in generative AI using backpropagation can be used to solve "inner" horn extension problems, but addressing "outer horn" extensions requires a more elaborate framework.

At the top level, GAIA uses the simplicial category of ordinal numbers with objects defined as $[n], n \geqslant 0$ and arrows defined as weakly order-preserving mappings $f : [n] \to [m]$, where $f(i) \leqslant f(j), i \leqslant j$. This top-level structure can be viewed as a combinatorial "factory" for constructing, manipulating, and destructing complex objects that can be built out of modular components defined over categories. The second layer of GAIA defines the building blocks of generative AI models as universal coalgebras over categories that can be defined using current generative AI approaches, including Transformers that define a category of permutation-equivariant functions on vector spaces, structured state-space models that define a category over linear dynamical systems, or image diffusion models that define a probabilistic coalgebra over ordinary differential equations. The third layer in GAIA is a category of elements over a (relational) database that defines the data over which foundation models are built. GAIA formulates the machine learning problem of building foundation models as extending functors over categories, rather than interpolating functions on sets or spaces, which yields canonical solutions called left and right Kan extensions. GAIA uses the metric Yoneda Lemma to construct universal representers of objects in non-symmetric generalized metric spaces. GAIA uses a categorical integral calculus of (co)ends to define two families of generative AI systems. GAIA models based on coends correspond to topological generative AI systems, whereas GAIA systems based on ends correspond to probabilistic generative AI systems.

Keywords Generative AI · Foundation Models · Higher-Order Category Theory · Machine Learning

*This is a preliminary draft of a forthcoming book. This draft may contain errors or omissions, and will be periodically updated.

Contents

1 Overview of the Paper

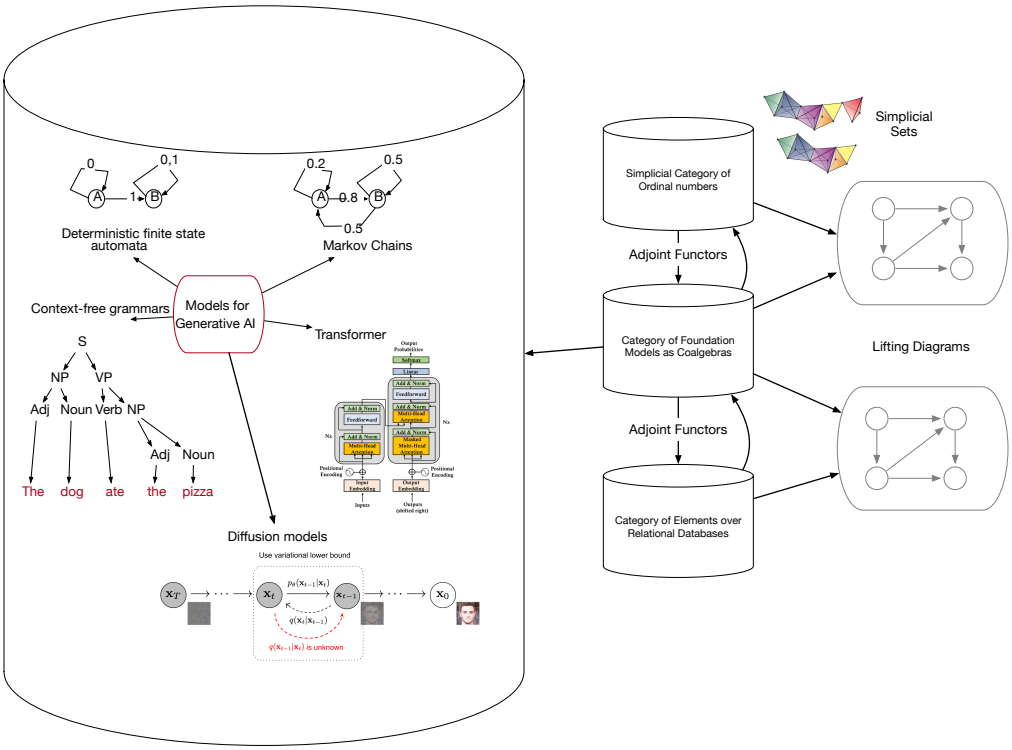


Figure 1: We propose a hierarchical Generative AI Architecture (GAIA) using higher-order category theory.

Generative AI has become a dominant paradigm for building intelligent systems in the last few years, ranging from large language models developed with the widely used Transformer model Vaswani et al. [2017], or more recently with the structured state space sequence models Gu et al. [2022], Yin et al. [2023], and with the growing use of image diffusion algorithms Song and Ermon [2019], Yin et al. [2023]. We can broadly define the problem of generative AI as the construction, maintenance, and deployment of foundation models Bommasani et al. [2022], a storehouse of human knowledge that provides the basic infrastructure for AI across some set of applications. A fundamental question, therefore, to investigate is to study the mathematical basis for foundation models. We propose a mathematical framework for a Generative AI Architecture (GAIA) (see Figure 1) based on the hypothesis that *category theory* MacLane [1971], Riehl [2017], Lurie [2009] provides a universal mathematical language for foundation models. In particular, GAIA is based on *simplicial learning*, which is intended to generalize *compositional* learning frameworks based on well-established machine learning algorithms, such as backpropagation Bengio [2009]. Category theory has been called a “Rosetta Stone" Baez and Stay [2010], as it provides a universal language for defining interactions among objects in all of mathematics, physics, computer science, and mathematical logic. Category theory has recently seen increasing use in machine learning, including dimensionality reduction McInnes et al. [2018] and clustering Carlsson and Memoli [2010]. One unique aspect of defining machine learning as extending functors in a category, in contrast to the well-established previous approach of extending functions over sets or spaces Wagstaff et al. [2022], is that there are two canonical solutions that emerge – the left and right Kan extensions MacLane [1971] – whereas there is no corresponding canonical solution to the problem of learning functions over sets.

Current generative AI systems are built on the longstanding algorithmic workhorse of backpropagation Bengio [2009]. Backpropagation is fundamentally a compositional sequential framework, where each module updates its parameters based on information it gets from its downstream modules, and in turn, transmits information back to upstream modules. Fong et al. [2019] propose a categorical framework for backpropagation, which we review in Section 2. In this paper, we will generalize this category-theoretic formalization of neural networks in several ways. As Figure 2 illustrates, unlike traditional generative AI models, such as those developed with sequence models like Transformers or structured state space sequence models, is not sequential, but rather *simplicial*. Each “face" of the n-simplex defines a “local" Transformer model (or indeed, any type of machine learning model), which are then “stitched" together into the whole structure using the mathematics of higher-order category theory of simplicial objects and sets May [1992].

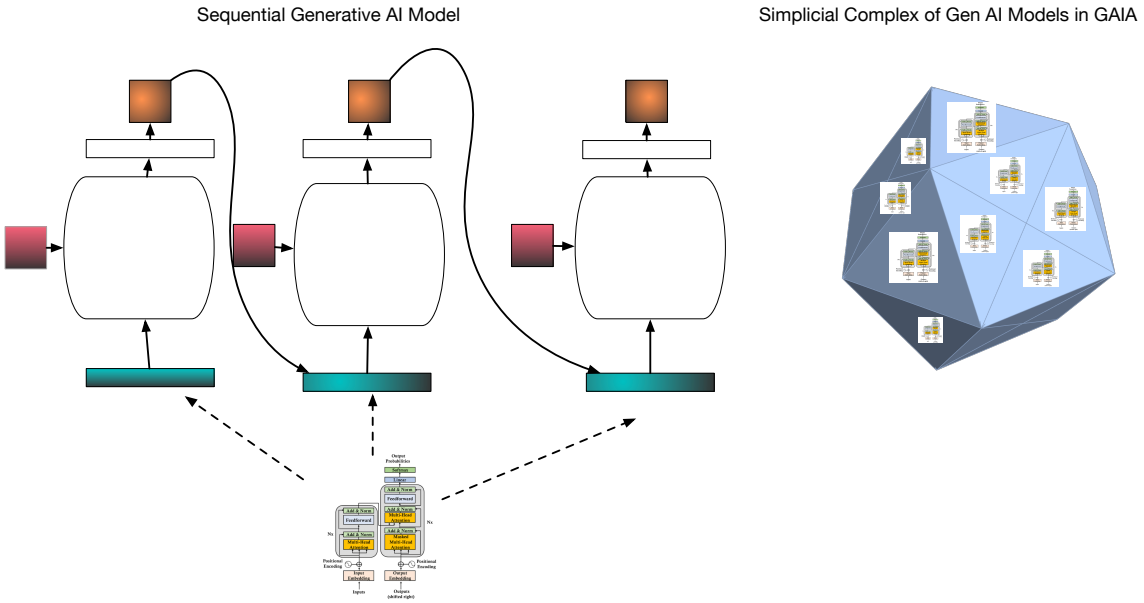


Figure 2: Traditional Generative AI models, such as Transformers, are based on a compositional sequential model. GAIA is based on a simplicial model, where each "face" of the n-simplicial complex defines a generative model.

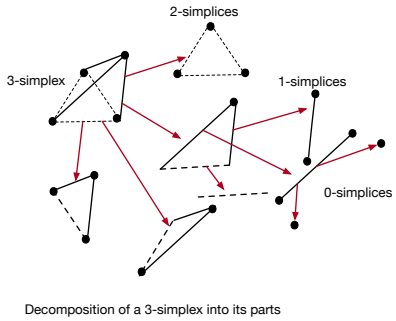


Figure 3: GAIA is based on a *hierarchical* framework, where each n-simplicial complex acts as a business unit in a company: each n-simplex updates its parameters based on data it receives from its superiors, and it transmits guidelines for its $n+1$ sub-simplicial complexes to help them with their updates. The mathematics for this hierarchical framework is based on higher-order category theory of simplicial sets and objects.

Figure 4 illustrates a crucial conceptual perspective that forms the basis for the design of GAIA. A generative model is, first and foremost, an *algebraic structure* of some kind. It may be a sequence, a directed graph, or as in our case, a simplicial complex. This specification is akin to specifying the "skeleton" of the generative AI model. To give the skeleton some "flesh and blood", it is necessary to map it into a parameter space (e.g., Euclidean space), through a suitable functor. The actual process of building a foundation model occurs through implementing a learning method, such as backpropagation, which Fong et al. [2019] model as a functor that maps from the space of parameters into the category of learners. A crucial difference in our approach is that we model backpropagation not just as a functor, but rather as an endofunctor on the category of parameters, as it must eventually result in a new set of parameters (see Figure 5. This important difference in our approach makes it possible to apply the rich theory of universal coalgebras over endofunctors Jacobs [2016], Rutten [2000] to analyze generative AI methods. We describe in detail in Section 2 one specific instantiation of this general perspective proposed by Fong et al. [2019]. In their case, the algebraic structure is a symmetric monoidal category that defines the skeleton. Their parameter space is Euclidean space, and their category of learners is defined by compositional learning using backpropagation. In GAIA, we make a more sophisticated framework, where the algebraic structure is a simplicial set or category, the parameter space may be a *sheaf* in a *topos* MacLane and leke Moerdijk [1994], and the category of learners is defined as horn extensions in a simplicial set.

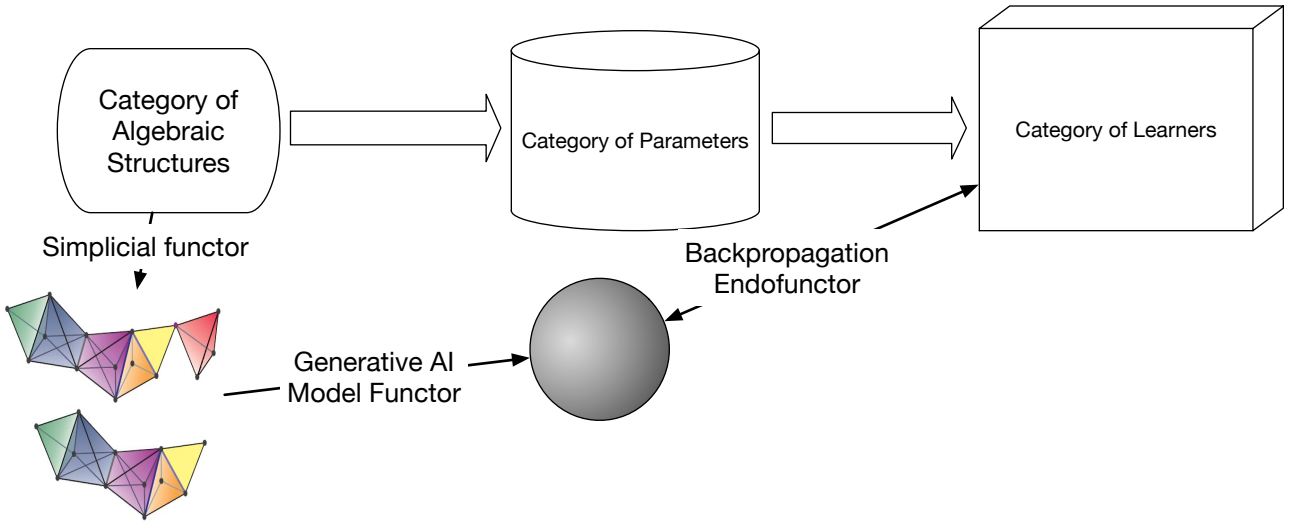


Figure 4: Crucial to the GAIA framework is understanding the separation between the algebraic structure of a generative AI model, and the parameter space over which the model is defined, and how specific machine learning algorithms such as backpropagation can be viewed as functors. Fong et al. [2019] defined backpropagation as a functor as shown from the category `Param` of parameters to the category `Learn` of machine learners. Crucially, GAIA models backpropagation as an *endofunctor* from the category `Param` back to itself, because every morphism in `Learn` must result in an update of the parameters of the network, thus resulting in a new object in `Param`. Thus, in this paper, we "close the loop", opening the rich theory of universal coalgebras defined by endofunctors Jacobs [2016], Rutten [2000] to analyze generative AI methods, such as backpropagation.

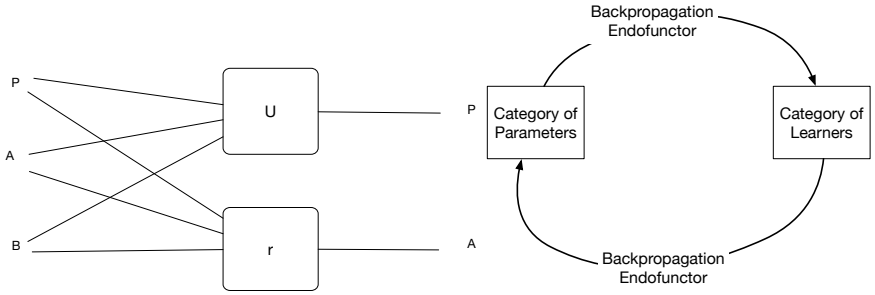


Figure 5: Left: In the categorical framework for deep learning proposed by Fong et al. [2019], a learner is a morphism in the category `Learn` that acts sequentially on its input A to produce an output B, updating its parameters P, and sending back a request A to an upstream module that represents "backpropagation". In GAIA, we view backpropagation as a *coalgebra* Rutten [2000], defined by an endofunctor on the category of parameters, so that each step of backpropagation is modeled as a dynamical system that maps some parameter object into a new parameter object.

GAIA is based on a hierarchical organization, much like the business units in a company. An n-simplex defines a collection of $n+1$ sub-simplicial sets (or object), and each n-simplex computes some function based on a set of parameters, which it updates based on information it receives from its superiors. It then transmits guidelines to its $n+1$ subordinate sub-simplicial sets on how to update their parameters. We use the simplicial category Δ at the top layer of GAIA to define not just sequences of morphisms, each representing a layer of a generative AI network, but simplicial complexes of them. One way to understand the connection between categories and simplicial sets is through the *nerve* functor that maps sequences of morphisms – for example, each representing a Transformer block or a diffusion image generation step – into a simplicial set. The n-simplices are defined by sequences of composable morphisms of length n. It can be shown that the nerve functor is a full and faithful functor that fully captures the category structure as a simplicial set Lurie [2009]. However, the left adjoint of the nerve functor is a "lossy" inverse, in that it only preserves structure for n-simplices, where $n \leqslant 2$. GAIA defines generative AI over n-simplicial complexes that allow more complex interactions among them than that which can be modeled by compositional learning frameworks, such as backpropagation. Simplicial sets are defined as a graded set $S_n, n \geqslant 0$, where S_0 represents "objects", S_1 represent morphisms (as in Fong et al. [2019]), S_2 define triangles of composable morphisms that have to be filled in different

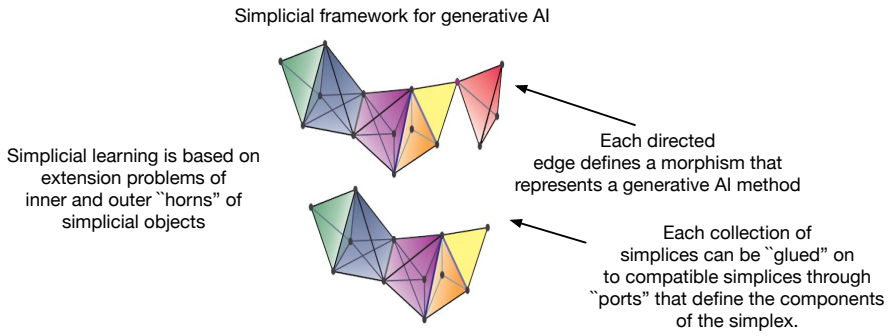
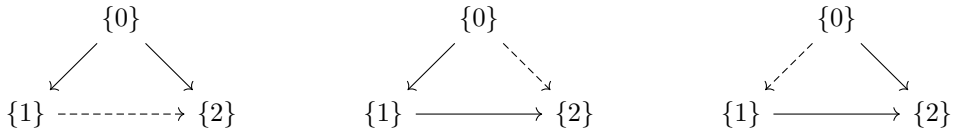


Figure 6: GAIA is based on a simplicial framework, where each generative AI method is modeled as a morphism that maps between two objects. In the simplest case of compositional learning, a 1-simplex is defined as an "edge", where its beginning and ending "vertices" represent data that flows into and out of a generative AI model, such as a Transformer, or a structured state space sequence model, or a diffusion process. Backpropagation can be used to solve compositional learning problems over such sequences of "edge" building blocks. GAIA generalizes this paradigm to define "higher-order" simplicial objects where the interfaces between simplices can be more elaborate. Each n-simplex is comprised of a family of $n-1$ subsimplicial objects, each of which can be "glued" together to form the n-simplex.

ways, and constitute "inner" and "outer" horn extension problems Lurie [2009]. In summary, GAIA generalizes the category-theoretic framework for deep learning in Fong et al. [2019] to a higher-order category of simplicial sets and objects. Figure 6 illustrates the simplicial generative AI vision underlying GAIA.

Backpropagation can solve a wide range of generative AI problems defined as supervised learning problems, where the task is to infer an unknown function $f : A \to C$ from samples $(a, f(a))$, where f is constructed as a sequential composition of building block unknown functions that represent intermediate targets such as $f \simeq g \circ h$, and $h : A \to B$, and $g : B \to C$. Such problems are modeled in GAIA as "inner horn" extensions of simplicial objects. GAIA is able to formulate "outer horn" extension problems, such as inferring unknown functions $f : B \to C$ from samples of unknown functions $g : A \to B$ and $h : A \to C$, or infer unknown functions $f : A \to B$ given samples of unknown functions $g : A \to C$ and $h : B \to C$, which lie outside the scope of sequential compositional methods like backpropagation. One example of a setting where both inner and outer horn extensions are solvable are in Kan complexes.

We can define the class of "horn extensions" of simplicial complexes, where each morphism might represent a generative AI morphism (such as in the category `Learn` considered by Fong et al. [2019]), which is essentially all the ways of composing 1-dimensional simplices to form a 2-dimensional simplicial object. Each simplicial subset of an n-simplex induces a a *horn* Λ^n_k, where $0 \leqslant k \leqslant n$. Intuitively, a horn Λ^n_k is a subset of a simplicial object that results from removing the interior of the n-simplex and the face opposite the kth vertex. Consider the three horns defined below. The dashed arrow $\dashrightarrow$ indicates edges of the 2-simplex Δ^2 not contained in the horns.



The inner horn Λ^2_1 is the middle diagram above, and admits an easy solution to the "horn filling" problem of composing the simplicial subsets. In defining a compositional category for neural networks and supervised learning, Fong et al. [2019] only consider "inner horn" extension problems defined as how to compose two morphism in the category `Learn`. In other words, if $f : A \to B$ and $g : B \to C$ are two functions to be learned from a database of samples, their framework works out the updates to the composition $g \circ f : A \to C$.

The two outer horns on either end pose a more difficult challenge. For example, filling the outer horn Λ^2_0 when the morphism between $\{0\}$ and $\{1\}$ is f and that between $\{0\}$ and $\{2\}$ is the identity $\mathbf{1}$ is tantamount to finding the left inverse of f up to homotopy. Dually, in this case, filling the outer horn Λ^2_2 is tantamount to finding the right inverse of f up to homotopy. A considerable elaboration of the theoretical machinery in category theory is required to describe the various solutions proposed, which led to different ways of defining higher-order category theory Boardman and Vogt [1973], Joyal [2002], Lurie [2009].

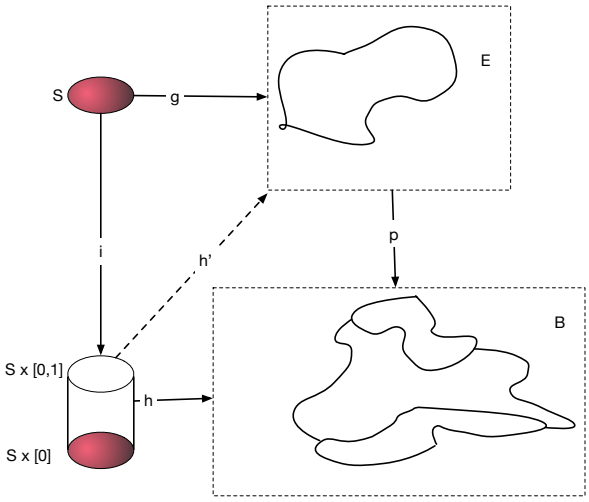


Figure 7: Lifting diagrams were originally studied in algebraic topology Gavrilovich [2017], and provide a concise way to define diverse computational problems in GAIA. A map $p : E \to B$ is called a *fibration* if and only if for any maps h and g that make this diagram "commute", there exists a diagonal map h' that makes the whole diagram commute. Fibrations have been used to formalize SQL queries in relational databases Spivak [2013] and causal inference Mahadevan [2023], and are central to homotopy theory in higher-order category theory. Many universal approximability results in deep learning Yarotsky [2018], Wagstaff et al. [2022], Yun et al. [2020] can be phrased in terms of lifting diagrams.

As examples, consider a large language model (LLM) that is trained to output programs based on textual inputs. For example, given data corresponding to possible textual descriptions of programs and actual code, a GitHub Copilot can generate programs from textual inputs. An outer horn problem for this case would be to infer an unknown function between two generated sample programs from their textual prompts as inputs. Similarly, for a generative AI program that produces images by diffusion from textual inputs, the outer horn problem might correspond to learning an unknown function between two generated images.

If we assume that the simplicial complex is a Kan complex Kan [1958], all horn extensions can be solved, which intuitively can be understood as implying that the outer horn extension problems can be turned into inner horn extensions. So, for example, we can solve the outer horn problem defined by the first diagram on the left above by assuming that the morphism $f : [0] \to [1]$ has an inverse $f^{-1} : [1] \to [0]$, and hence turn the outer horn into an inner horn problem. Similarly, for the outer horn problem on the right hand side of the above diagram, we can assume that morphism $f : [0] \to [2]$ has an inverse $f^{-1} : [2] \to [0]$ that converts it back into an inner horn extension problem.

Note that neural networks that are trained through backpropagation are inherently *directional*: there is a well-defined notion of an input and an output over which forward and backwards propagation occurs. In essence, what outer horn extension problems imply is that if there exists a solution to a problem of inferring an unknown function $f : A \to B$ from samples $(a, f(a)) \in A \times B$, does that imply a solution to the problem of inferring an inverse function $f^{-1} : B \to A$? Lastly, it must be noted that horn extensions can be more complex than the simple 2-simplex case described above. In general, in a lifting diagram, defined below, we are asking if a solution exists to an arbitrary lifting problem in a certain category?

We define the update process through lifting diagrams from algebraic topology Gavrilovich [2017] as a unifying framework, from answering queries to building foundation models. A lifting diagram defines constraints between different paths that lead from one category to another. They have been used to formulate queries in relational databases Spivak [2013]. In our previous work, we used lifting diagrams to define queries for causal inference Mahadevan [2023]. Lifting problems define ways of decomposing structures into simpler pieces, and putting them back together again, and thus play a central role in GAIA (see Figure 7.

Definition 1. Let $\mathcal{C}$ be a category. A **lifting problem** in $\mathcal{C}$ is a commutative diagram σ in $\mathcal{C}$.

$$\begin{array}{ccc} A & \xrightarrow{\mu} & X \\ \downarrow{\scriptstyle f} & & \downarrow{\scriptstyle p} \\ B & \xrightarrow{\nu} & Y \end{array}$$

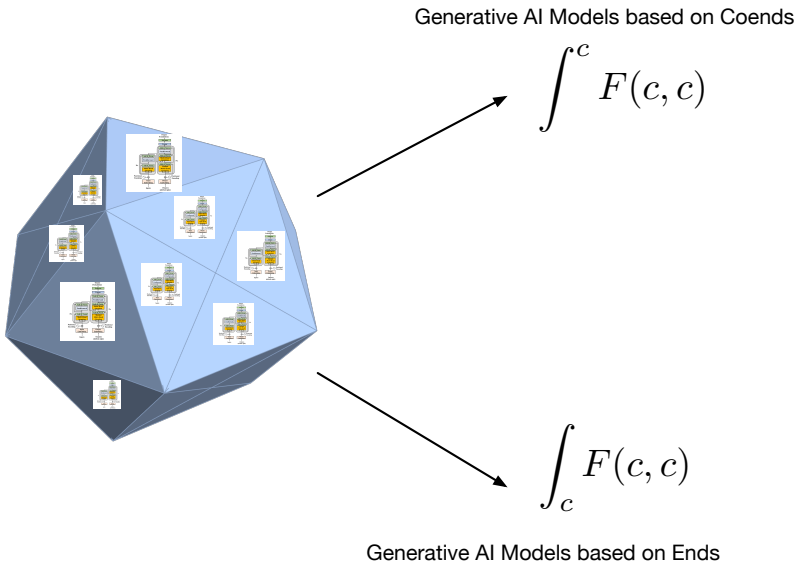


Figure 8: We propose two families of GAIA models in this paper (see Section 7 for details), based on coends and ends Loregian [2021]. In this diagram, the bifunctor $F \in \text{Cat}(\mathcal{C}^{op} \times \mathcal{C}, \mathcal{D})$ acts both contravariantly and covariantly on objects in the category $\mathcal{C}$. Coend and end objects correspond to objects in the category $\mathcal{D}$. Coend GAIA models are based on topological realizations of the simplicial model, whereas end GAIA models are based on probabilistic generative models.

To understand the meaning of such a diagram for generative AI, let us consider the setting where every edge in the above commutative diagram represents an instance of a Transformer module that maps a object $x \in \mathbb{R}^{d \times n}$ into another using a permutation-equivariant mapping. Yun et al. [2020] show that Transformers compute permutation-equivariant functions and are nonetheless universal approximators in the space of all continuous functions on $\mathbb{R}^{d \times n}$ due to their reliance on absolute positional encoding Vaswani et al. [2017] to overcome the limitations imposed by permutation equivariance. Permutation-equivariant functions are defined as $f : \mathbb{R}^{d \times n} \to \mathbb{R}^{d \times n}$ such that $f(XP) = f(X)P$ for any $X \in \mathbb{R}^{d \times n}$ where P is a permutation matrix. It is straightforward to define a category of Transformer models, where the objects are vectors $X \in \mathbb{R}^{d \times n}$ and the arrows are permutation-invariant mappings. Similarly, diffusion models used in image generation Song and Ermon [2019] can be viewed as a category of stochastic dynamical systems that can be viewed as probabilistic coalgebras Sokolova [2011]. Rutten [2000] and Jacobs [2016] show that a wide class of dynamical systems used in computer science can be expressed as categories of universal coalgebras. With this context, asking for solutions to lifting problems is posing a question on the representational adequacy of a framework for generative AI.

Definition 2. Let $\mathcal{C}$ be a category. A **solution to a lifting problem** in $\mathcal{C}$ is a morphism $h : B \to X$ in $\mathcal{C}$ satisfying $p \circ h = \nu$ and $h \circ f = \mu$ as indicated in the diagram below.

$$\begin{array}{ccc} A & \xrightarrow{\mu} & X \\ \downarrow{\scriptstyle f} & \overset{h}{\nearrow} & \downarrow{\scriptstyle p} \\ B & \xrightarrow{\nu} & Y \end{array}$$

Although lifting diagrams have been proposed for deep learning architectures recently Papillon et al. [2023], it is important to stress that the notion of simplicial complex used in this paper, as well as other notions used in computer graphics is fundamentally different from our paper. In our case, simplicial complexes are directional, since each edge is defined by a directional arrow, not an undirected arrow. The edges correspond to morphisms, and indeed, a basic question that we will ask is under what circumstances can directional morphisms be inverted. This question is fundamental to solving extension problems, and in really nice settings like Kan complexes, all morphisms can be inverted since all inner and outer horn extension problems have solutions.

In the literature on higher-order category theory Joyal [2002], Lurie [2009, 2022] and homotopy theory Gabriel et al. [1967], Richter [2020], Quillen [1967], lifting diagrams were used to define structures such as Kan complexes that possess nice extension properties. For example, for any given topological space X, there is a functor that defines a simplicial set $\text{Sing}_\bullet(X)$ defined by all continuous functions from the topological simplex Δ_n into the space X. The

simplicial set $\text{Sing}_\bullet(X)$ is a Kan complex because all "horn extensions" of simplicial subsets can be solved. In practical terms, this property explains the success of dimensionality reduction methods like UMAP McInnes et al. [2018], which constructs functors from simplicial sets that represent data into topological spaces. The topological realization of simplicial sets computed by UMAP is an example of a coend, a unifying "integral calculus" proposed originally by Yoneda Yoneda [1960]. Loregian [2021] gives a detailed description of the integral calculus of (co)ends, which provides a unifying way to design an entire spectrum of generative AI systems, where systems based on ends leads to probabilistic generative models like Transformers, whereas models based on coends lead to topologically based generative AI systems that have not been explored in the literature.

In this paper, we use the framework of lifting diagrams to both formulate queries as shown in Figure 1, but also to define the problem of building foundation models from data. In particular, we build on the framework of simplicial sets and objects, where we pose lifting diagram queries as solving "horn extension" problems Lurie [2009]. Formally, we pose the problem of learning in a generative AI system in terms of properties such as Kan complexes May [1992], which are ideal categories to solve lifting problems since there is a unique solution to all extension problems (inner and outer horn extensions). As a concrete example, every topological space can be mapped into a simplicial set using the singular functor, which was used in the UMAP McInnes et al. [2018] dimensionality reduction method, which can be shown to form a Kan complex.

Beginning at the top layer, GAIA uses the simplicial category of ordinal numbers May [1992] as a way to build, manipulate, and destroy compositional structures. The category of ordinal numbers Δ includes a collection of objects indexed by the ordinals $[n], n \geqslant 0$, where $[n] = (0, 1, \ldots, n)$ under the natural ordering $<$. The morphisms of Δ are order-preserving functions $f : [n] \to [m]$ where $f(i) \leqslant f(j), i \leqslant j, i, j \in [n]$. The category Δ has provided the basis for a combinatorial approach to topology and also serves as the basis for higher-order category theory Joyal [2002], Lurie [2009], Gabriel et al. [1967]. This category "comes to life" when it is functorially mapped to some other category, such as **Sets**, when the resulting structure is called a simplicial set. The contravariant functor $X : \Delta^{op} \to \mathbf{Sets}$ is defined by viewing $X([0])$ as a set of "objects", $X[1])$ as a set of "arrows" representing pairwise interactions among the objects, and in general, $X([n])$ – which is often simply written as X_n and consists of a set of n-simplices – defines interactions of order n among the objects. Simplicial sets generalize directed graphs, partial orders, sequences, and in fact, regular categories MacLane [1971] as well. There are constructor and destructor morphisms that map from $X([n])$ to $X([n+1])$ and $X([n])$ to $X(n-1])$, which are usually denoted as degeneracy and face operators.

The second layer of GAIA defines a category of generative AI models, which can be composed of any of the standard technologies used to build generative models, including finite and probabilistic automata, context-free grammars, structured state-space sequence models Gu et al. [2022] or Transformer models Vaswani et al. [2017], or cellular automata Vollmar [2006], Wolfram [2002]. We assume that each of these models defines a category whose objects and arrows can be composed and otherwise manipulated by the simplicial category Δ.

The third layer of GAIA defines the category of (relational) databases out of which the category of foundation models is build (e.g., such as by using one of the standard generative AI methods, such as self-attention Vaswani et al. [2017] or structured state-space sequence models Gu et al. [2022]). Spivak and Kent [2012] has shown that categories provide a foundation for defining relational databases, and that many common operations in databases can be defined in terms of lifting diagrams in topology Gavrilovich [2017]. In particular, a fundamental premise of GAIA is that machine learning is defined as the extension of *functors* on categories, not functions on sets Wagstaff et al. [2022]. The fundamental reason to view machine learning as extending functors is that there are two canonical solutions to the problem of extending functors, defined as left and right Kan extensions Kan [1958]. In contrast, there is no obvious or natural solution to the problem of extending functions on sets, which has prompted an enormous literature in the field of machine learning over many decades, and also in fields like information theory Chaitin [2002], Cover and Thomas [2006]. Lifting diagrams provide an elegant and general framework to pose the problem of generalization for generative AI, based not just on individual units of experience, but by providing a theoretically sound way to do generalization over arbitrary relational structures. Much of the work in machine learning has focused on the ability to generalize propositional representations, which also includes most of the work in statistics. To generalize over first-order relational structures requires bringing in some powerful tools from algebraic topology and higher-order category theory, in particular the ability to do *horn filling* of simplicial horns Lurie [2009].

1.1 Roadmap to the Paper

Given the length of this paper, a roadmap to its organization will be helpful to the reader. Keep in mind that this paper is a condensed version of a forthcoming book, which is designed to provide a detailed tutorial level introduction to category theory, in addition to illustrating its application to generative AI. With that mind, Section 2 begins us off with a detailed look at a category theory of deep learning, building on the work of Fong et al. [2019]. The crucial idea of separating the algebraic structure of a generative AI model from its parameterization, which in turn is independent of

the structure of a learning framework is crucial to our framework as well, although GAIA differs in many ways from the approach proposed in Fong et al. [2019].

Section 3 gives our alternative view of backpropagation as an endofunctor, in particular a universal coalgebra of the form $X \to F(X)$, where the endofunctor F maps objects X in a category $\mathcal{C}$ back to the same category. Universal coalgebras Jacobs [2016], Rutten [2000] provide a rich language for specifying dynamical systems, and they have also been extended to describe probabilistic generative models, such as Markov chains Sokolova [2011], Markov decision processes Feys et al. [2018] and a wealth of programming-related abstractions Jacobs [2016].

Section 4 defines the simplicial layer of GAIA, which acts like a "combinatorial factory" that can assemble together pieces of generative AI models. The heart of the GAIA framework is that the simplicial category allows a hierarchical framework for generative AI, which we believe goes beyond the purely sequential framework thus far studied in the literature. We illustrate how hierarchical learning works in GAIA in terms of lifting problems in simplicial sets. Section 5 defines particular categories for generative AI, including the popular Transformer architecture as permutation equivariant functions over Euclidean spaces. Section 6 defines universal properties and the Yoneda Lemma, which are used to define universal parameterizations of generative AI models. In particular, we show that non-symmetric generalized metric spaces can be studied with the metric Yoneda Lemma, which has applications in constructing non-symmetric attention models for natural language.

Section 7 defines an abstract integral calculus for generative AI, based on Yoneda's pioneering work Yoneda [1960]. Loregian [2021] gives a detailed textbook level account of (co)end calculus. We define two classes of generative AI models, those based on coends and ends. We show that coend GAIA models lead to topological realizations, whereas GAIA models based on ends lead to probabilistic generative AI models. We also introduce sheaves and topoi as alternative parameterizations of generative AI models, which arise from the Yoneda Lemma, and can give additional structure over simply using Euclidean spaces. Section 8 finally defines abstract notions of equivalence in category theory for comparing two generative AI models. When can we say, for example, that a summarized document produced by a generative AI copilot is actually faithful to the original document on which it was based? Homotopy theory provides some answers to these questions, which have only been studied empirically in the literature. We introduce the notion of a *classifying space* for generative AI models, and define their homotopy colimits.

The paper covers a great deal of abstract mathematics, but we have attempted to provide a range of concrete examples of its application to the problem of generative AI. Many more examples can be given, but would significantly increase the length of an already really long paper! Ultimately, as we conclude at the end, the real proof of the utility of GAIA will come from its actual implementation as a working system, but we view that as a multi-year research problem. There are many open problems that are remaining to be worked out, and we discuss a few of them in the paper at various places.

2 Backpropagation as a Functor: Compositional Learning

Our principal goal in this section to review the categorical framework for deep learning proposed in Fong et al. [2019], which models backpropagation as a functor. In the next section, we will argue that backpropagation should be viewed instead as an *endufunctor* on the category *Param*, which defines the space over which generative AI model are defined.

We first give a high-level introduction to generative AI, building on the framework of category theory (see Figure 9). Category theory is intrinsically a framework for compositional structures, which generative AI exemplifies as well. Excellent textbook length treatments are readily available and should be consulted for further background MacLane [1971], MacLane and leke Moerdijk [1994], Riehl [2017], Richter [2020]. We summarize salient concepts from category theory as and when needed. We define several types of categories in this section, beginning with a category for supervised learning, and then other categories that represent machine learning algorithms, including the traditional backpropagation algorithm, and zeroth-order optimization, as well as categories for specific deep learning architectures, such as Transformers, structured state space sequence models, and diffusion models. We then introduce some key ideas from category theory, including the fundamental Yoneda Lemma that shows all objects in a category can be characterized in terms of their interactions.

2.1 Category of Supervised Learning

Fong et al. [2019] give an elegant characterization of the well-known backpropagation algorithm that serves as the "workhorse" of deep learning as a functor over symmetric monoidal categories. In such categories, objects can be "multiplied": for example, sets form a symmetric monoidal category as the Cartesian product of two sets defines a multiplication operator. A detailed set of coherence axioms are defined for monoidal categories (see MacLane [1971] for details), which we will not go through, but they ensure that multiplication is associative, as well as that there are identity operators such that $I \otimes A \simeq A$ for all objects A, where I is the identity object.

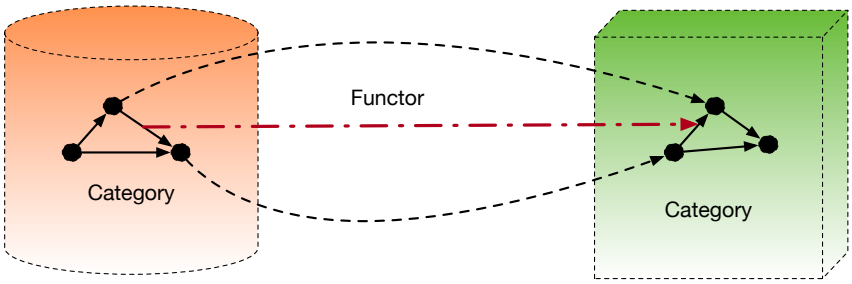


Figure 9: Categories are defined by collection of arbitrary objects that interact through morphisms (also called arrows). Functors map objects from one category into another, but also map the arrows of the domain category into corresponding arrows in the co-domain category. We define generative AI systems and learning algorithms as categories in GAIA.

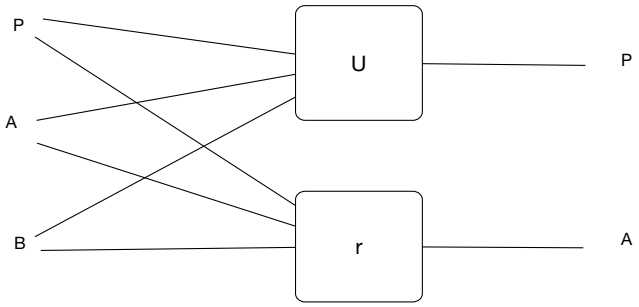


Figure 10: A learner in the symmetric monoidal category `Learn` is defined as a morphism. Later in Section 3, we will see how to define learners as coalgebras instead.

Definition 3. Fong et al. [2019] The symmetric monoidal category **Learn** is defined as a collection of objects that define sets, and a collection of an equivalence class of learners. Each learner is defined by the following 4-tuple (see Figure 10).

- A parameter space P
- An implementation function $I : P \times A \to B$
- An update function $U : P \times A \times B \to P$
- A request function $r : P \times A \times B \to A$

Note that it is the request function that allows learners to be composed, as each request function transmits information back upstream to earlier learners what output they could have produced that would be more “desirable". This algebraic characterization of the backpropagation algorithm clarifies its essentially compositional nature

Two learners (P, I, U, R) and (P', I', U', r') are equivalent if there is a bijection $f : P \to P'$ such that the following identities hold for each $p \in P, a \in A$ and $b \in B$.

- $I'(f(p), a) = I(p, a)$.
- $U'(f(p), a, b) = f(U(p, a, b))$.
- $r'(f(p), a, b) = r(p, a, b)$

Typically, in generative AI trained with neural networks, the parameter space $P = \mathbb{R}^N$ where the neural network has N parameters. The implementation function I represents the “feedforward" component, and the request function represents the “backpropagation" component. The update function represents the change in parameters as a result of processing a training example $(a, f(a)) \in A \times B$. The main contribution of Fong et al. [2019] is in showing that supervised learning can be defined as a compositional category under the sequential composition of morphisms defining individual building blocks of learners.

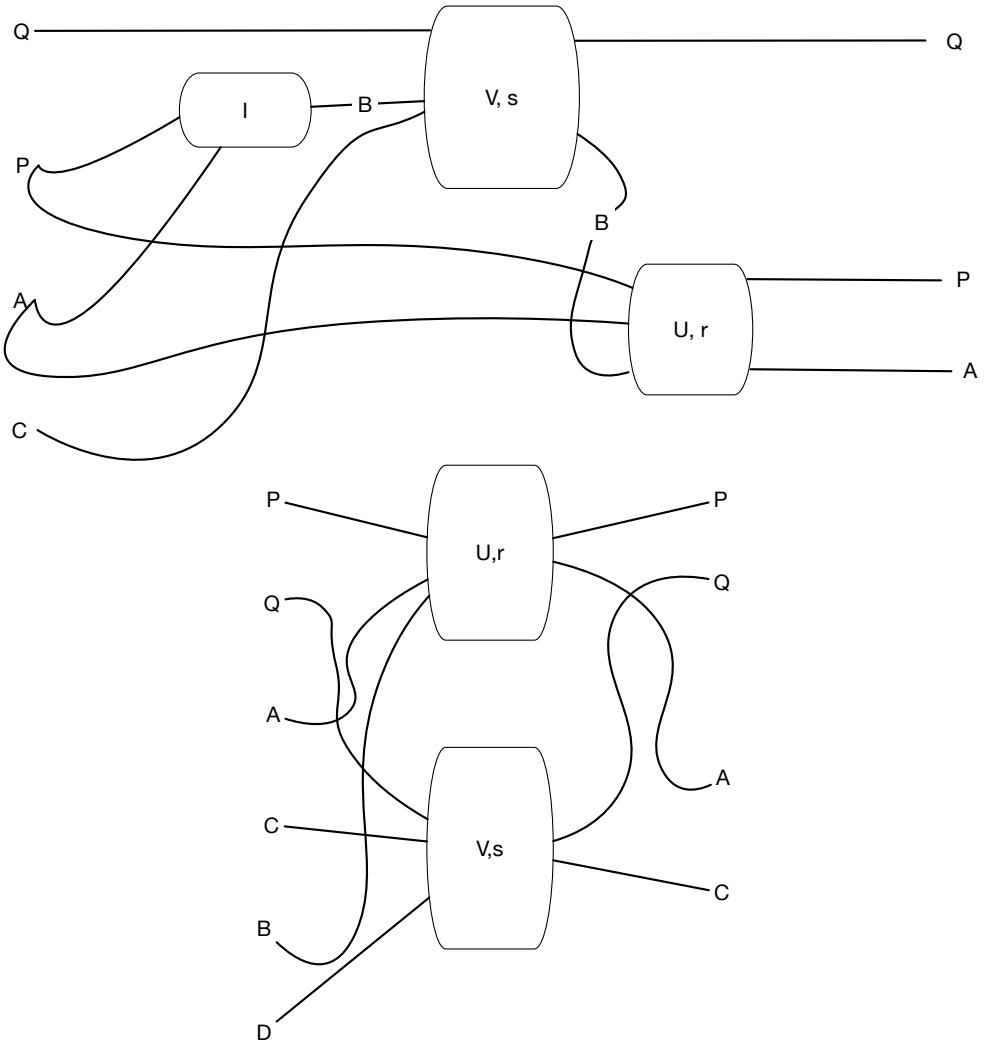


Figure 11: Sequential and parallel composition of two learners in the symmetric monoidal category `Learn`.

Fong et al. [2019] show that each learner can be combined in sequentially and in parallel (see Figure 11), both formally using the operations of composition $\circ$ and tensor product $\otimes$ in the symmetric monoidal category `Learn`, and equivalently in terms of string diagrams. For clarity, let us write out the compositional rule for a pair of learners

$$A \xrightarrow{(P,I,U,r)} B \xrightarrow{(Q,J,V,s)} C$$

The composite learner $A \to C$ is defined as $(P \times Q, I \cdot J, U \cdot V, r \cdot s)$, where the composite implementation function is

$$(I \cdot J)(p, q, a) \coloneqq J(q, I(p, a))$$

and the composite update function is

$$U \cdot V(p, q, a, c) \coloneqq \left(U(p, a, s(q, I(p, a), c)), V(q, I(p, a), c)\right.$$

and the composite request function is

$$(r \cdot s)(p, q, a, c) \coloneqq r(p, a, s(q, I(p, a), c)).$$

2.2 Backpropagation as a Functor

We can define the backpropagation procedure as a functor that maps from the category `Para` to the category `Learn`. Functors can be viewed as a generalization of the notion of morphisms across algebraic structures, such as groups, vector spaces, and graphs. Functors do more than functions: they not only map objects to objects, but like graph homomorphisms, they need to also map each morphism in the domain category to a corresponding morphism in the co-domain category. Functors come in two varieties, as defined below. The Yoneda Lemma, in its most basic form, asserts that any set-valued functor $F : \mathcal{C} \to \mathbf{Sets}$ can be universally represented by a *representable functor* $\mathcal{C}(-, x) : \mathcal{C}^{op} \to \mathbf{Sets}$.

Definition 4. A **covariant functor** $F : \mathcal{C} \to \mathcal{D}$ from category $\mathcal{C}$ to category $\mathcal{D}$, and defined as the following:

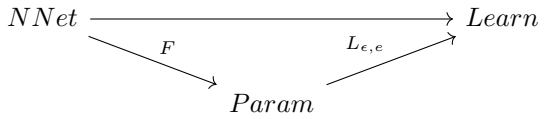
- An object FX (also written as $F(X)$) of the category $\mathcal{D}$ for each object X in category $\mathcal{C}$.
- An arrow $F(f) : FX \to FY$ in category $\mathcal{D}$ for every arrow $f : X \to Y$ in category $\mathcal{C}$.
- The preservation of identity and composition: $F\ id_X = id_{FX}$ and $(Ff)(Fg) = F(g \circ f)$ for any composable arrows $f : X \to Y, g : Y \to Z$.

Definition 5. A **contravariant functor** $F : \mathcal{C} \to \mathcal{D}$ from category $\mathcal{C}$ to category $\mathcal{D}$ is defined exactly like the covariant functor, except all the arrows are reversed.

The *functoriality* axioms dictate how functors have to be behave:

- For any composable pair f, g in category C, $Fg \cdot Ff = F(g \cdot f)$.
- For each object c in C, $F(1_c) = 1_{Fc}$.

Note that the category `Learn` is ambivalent as to what particular learning method is used. To define a particular learning method, such as backpropagation, we can define a category whose objects define the parameters of the particular learning method, and then another category for the learning method itself. We can define a functor from the category `NNet` to the category `Learn` that factors through the category `Param`. Later in the next section, we show how to generlize this construction to simplicial sets.



Definition 6. Fong et al. [2019] The category `Param` defines a strict symmetric monoidal category whose objects are Euclidean spaces, and whose morphisms $f : \mathbb{R}^n \to \mathbb{R}^m$ are equivalence classes of differential parameterized functions. In particular, (P, I) defines a Euclidean space P and $I : P \times A \to B$ defines a differentiable parameterized function $A \to B$. Two such pairs $(P, I), (P', I')$ are considered equivalent if there is a differentiable bijection $f : P \to P'$ such that for all $p \in P$, and $a \in A$, we have that $I'(f'(p), a) = I(p, a)$. The composition of $(P, I) : \mathbb{R}^n \to \mathbb{R}^m$ and $(Q, J) : \mathbb{R}^n \to \mathbb{R}^m$ is given as

$$(P \times Q, I \cdot J) \quad \text{where} \quad (I \cdot J)(p, q, a) = J(q, I(p, a))$$

The monoidal product of objects $\mathbb{R}^n$ and $\mathbb{R}^m$ is the object $\mathbb{R}^{n+m}$, whereas the monoidal product of morphisms $(P, I) : \mathbb{R}^m \to \mathbb{R}^m$ and $(Q, J) : \mathbb{R}^l \to \mathbb{R}^k$ is given as $(P \times Q, I \parallel J)$, where

$$(I \parallel J)(p, q, a, c) = (I(p, a), J(q, c))$$

Symmetric monoidal categories can also be braided. In this case, the braiding $\mathbb{R}^m \parallel \mathbb{R}^m \to \mathbb{R}^m \parallel \mathbb{R}^n$ is given as $(\mathbb{R}^0, \sigma)$ where $\sigma(a, b) = (b, a)$.

The backpropagation algorithm can itself be defined as a functor over symmetric monoidal categories

$$L_{\epsilon,e} : \texttt{Param} \to \texttt{Learn}$$

where $\epsilon > 0$ is a real number defining the learning rate for backpropagation, and $e(x, y) : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ is a differentiable error function such that $\frac{\partial e}{\partial x}(x_0, -)$ is invertible for each $x_0 \in \mathbb{R}$. This functor essentially defines an update procedure

for each parameter in a compositional learner. In other words, the functor $L_{\epsilon,e}$ defined by backpropagation sends each parameterized function $I : P \times A \to B$ to the learner (P, I, U_I, r_I)

$$U_I(p, a, b) \coloneqq p - \epsilon \nabla_p E_I(p, a, b)$$

$$r_I(p, a, b) \coloneqq f_a(\nabla_a E_I(p, a, b))$$

where $E_I(p, a, b) \coloneqq \sum_j e(I_j(p, a), b_j)$ and f_a is a component-wise application of the inverse to $\frac{\partial e}{\partial x}(a_i, -)$ for each i.

Note that we can easily define functors that define other ways of doing parameterized updates, such as a stochastic approximation method Robbins and Monro [1951] that updates each parameter using only the (noisy) value of the function at the current value of the parameter, and uses a gradual decay of the learning parameters to "simulate" the process of taking gradients. These sort of stochastic approximation updates are now called "zeroth-order" optimization in the deep learning literature.

3 Backpropagation as an Endofunctor: Generative AI using Universal Coalgebras

Our categorical framework for generative AI differs in crucial ways from the analysis in Fong et al. [2019] that defined backpropagation as a functor, but not an endofunctor. In their framework, which we reviewed in the previous section, backpropagation was modeled as a functor from the category `Param` to the cateory `Learn`, but that masks the simple fact that the goal of learning is to produce a new set of parameters (i.e., construct a new object in `Param`). Once you complete that loop, backpropagation becomes an endofunctor. This property allows bringing in the rich framework of universal coalgebras Jacobs [2016], Rutten [2000] to analyze a whole family of endofunctors for generative AI.

As the ultimate goal of backpropagation at each step is to produce a new parameter, i.e. a new object in `Param`, we argue that our endofunctor characterization provides a rich source of insight into the analysis of generative AI methods. Accordingly, we review below the theory of universal coalgebras, and then show more formally how to model backpropagation and other similiar generative AI methods as coalgebras.

3.1 Non-Well-Founded Sets and Universal Coalgebras

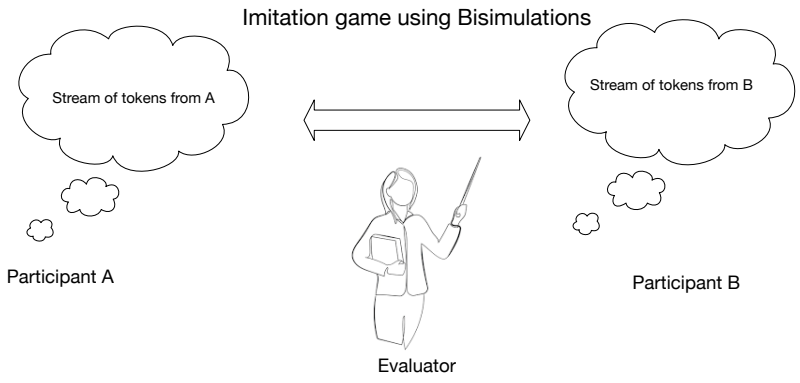


Figure 12: Generative models define an infinite stream of tokens. Solving an 'imitation game" Turing [1950] that involves comparing two infinite data streams involves the process of deciding if two non-well-founded sets are categorically *bisimulations* of each otherAczel [1988], Rutten [2000].

To begin with, we present an elegant formalism for defining generative AI models as universal coalgebras Rutten [2000], and non-well-founded sets Aczel [1988]. Figure 12 illustrates the main idea. We define the two participants in an imitation game as universal coalgebras (or non-well-founded sets) and ask if there is a bisimulation relationship between

them. This characterization covers a wide range of probabilistic models, including Markov chains and Markov decision processes Sutton and Barto [1998], and automata-theoretic models, as well as generative AI models Gu et al. [2023].

Generative AI has become popular recently due to the successes of neural and structured-state space sequence models Gu et al. [2022], Vaswani et al. [2017] and text-to-image diffusion models Song and Ermon [2019]. The underlying paradigm of building generative models has a long history in computer science and AI, and it is useful to begin with the simplest models that have been studied for several decades, such as deterministic finite state machines, Markov chains, and context-free grammars. We use category theory to build generative AI models and analyze them, which is one of the unique and novel aspects of this paper. To explain briefly, we represent a generative model in terms of *universal coalgebras* Rutten [2000] generated by an *endofunctor* F acting on a category C. Coalgebras provide an elegant way to model dynamical systems, and capture the notion of state Jacobs [2016] in ways that provide new insight into the design of AI and ML systems. Perhaps the simplest and in some ways, the most general, type of generative AI model that is representable as a coalgebra is the *powerset functor*

$$F : S \Rightarrow \mathcal{P}(S)$$

where S is any set (finite or not), and $\mathcal{P}(S)$ is the set of all subsets of S, that is:

$$\mathcal{P}(S) = \{A | A \subseteq S\}$$

Notice in the specification of the powerset functor coalgebra, the same term S appears on both sides of the equation. That is a hallmark of coalgebras, and it is what distinguishes coalgebras from algebras. Coalgebras generate search spaces, whereas algebras compact search spaces and summarize them. This admittedly simple structure nonetheless is extremely versatile and enables modeling a remarkably rich and diverse set of generative AI models, including the ones listed in Figure 13. To explain briefly, we can model a context-free grammar as a mapping from a set S that includes all the vertices in the context-free grammar graph shown in Figure 13 to the powerset of the set S. More specifically, if $S = N \cup T$ is defined as the non-terminals N as well as the terminal symbols (the actual words) T, any context-free grammar rule can be represented in terms of a power set functor. We will explain how this approach can be refined later in this Section, and in much more detail in later sections of the paper. To motivate further why category theory provides an elegant way to model generative AI systems, we look at some actual examples of generative AI systems to see why they can be modeled as functors.

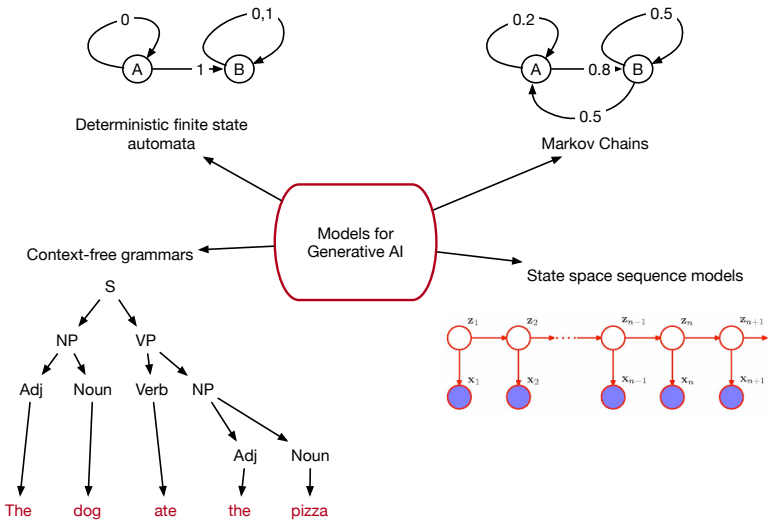


Figure 13: In this paper, generative AI models, from the earliest models studied in computer science such as deterministic finite state automata and context-free grammars, to models in statistics and information theory like Markov chains, and lastly, sequence models can be represented as universal coalgebras.

To compare say two large language models, we need to compare two potentially infinite data streams of tokens (e.g., words, or in general, other forms of communication represented digitally by bits). Many problems in AI and ML involve reasoning about circular phenomena. These include reasoning about *common knowledge* Barwise and Moss [1996], Fagin et al. [1995] such as social conventions, dealing with infinite data structures such as lists or trees in

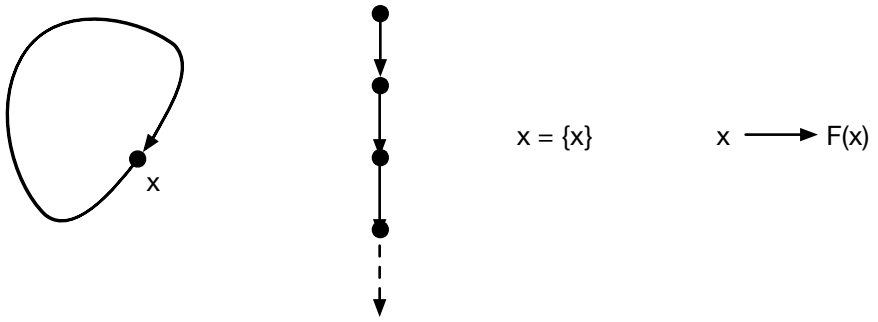


Figure 14: Three representations of infinite data streams: non-well-founded set $x = \{x\}$: accessible pointed graphs (AGPs), non-well-founded sets specified by systems of equations, and universal coalgebras. We can view these as generative models of the recursive set $\{\{\{\ldots\}\}\}$.

computer science, and causal inference in systems with feedback where part of the input comes from the output of the system. In all these situations, there is an intrinsic problem of having to deal with infinite sets that are recursive and violate a standard axiom called well-foundedness in set theory. First, we explain some of the motivations for including non-well-founded sets in AI and ML, and then proceed to define the standard ZFC axiomatization of set theory and how to modify it to allow circular sets. We build on the pioneering work of Peter Aczel on the anti-foundation-axiom in modeling non-well-founded sets Aczel [1988], which has elaborated previously in other books as well Barwise and Moss [1996], Jacobs [2016], although we believe this paper is perhaps one of the first to focus on the application of non-well-founded sets and universal coalgebras to problems in AI and ML at a broad level.

Figure 14 illustrates three ways to represent an infinite object, as a directed graph, a (non-well-founded) set or as a system of equations. We begin with perhaps the simplest approach introduced by Peter Aczel called accessible pointed graphs (APGs) (see Figure 14), but also include the category-theoretic approach of using *universal coalgebras* Rutten [2000], as well as systems of equations Barwise and Moss [1996].

We now turn to describing coalgebras, a much less familiar construct that will play a central role in the proposed ML framework of coinductive inference. Coalgebras capture hidden state, and enable modeling infinite data streams. Recall that in the previous Section, we explored non-well-founded sets, such as the set $\Omega = \{\Omega\}$, which gives rise to a circularly defined object. As another example, consider the infinite data stream comprised of a sequence of objects, indexed by the natural numbers:

$$X = (X_0, X_1, \ldots, X_n, \ldots)$$

We can define this infinite data stream as a coalgebra, comprised of an accessor function **head** that returns the head of the list, and a destructor function that gives the **tail** of the list, as we will show in detail below.

To take another example, consider a deterministic finite state machine model defined as the tuple $M = (X, A, \delta)$, where X is the set of possible states that the machine might be in, A is a set of input symbols that cause the machine to transition from one state to another, and $\delta : X \times A \to X$ specifies the transition function. To give a coalgebraic definition of a finite state machine, we note that we can define a functor $F : X \to \mathcal{P}(A \times X)$ that maps any given state $x \in X$ to the subset of possible future states y that the machine might transition to for any given input symbol $a \in A$.

We can now formally define F-coalgebras analogous to the definition of F-algebras given above.

Definition 7. Let $F : \mathcal{C} \to \mathcal{C}$ be an endofunctor on the category $\mathcal{C}$. An **F-coalgebra** is defined as a pair (A, α) comprised of an object A and an arrow $\alpha : A \to F(A)$.

The fundamental difference between an algebra and a coalgebra is that the structure map is reversed! This might seem to be a minor distinction, but it makes a tremendous difference in the power of coalgebras to model state and capture dynamical systems. Let us use this definition to capture infinite data streams, as follows.

$$\mathbf{Str} : \mathbf{Set} \to \mathbf{Set}, \qquad \mathbf{Str}(X) = \mathbb{N} \times X$$

Here, **Str** is defined as a functor on the category **Set**, which generates a sequence of elements. Let N^ω denote the set of all infinite data streams comprised of natural numbers:

$$N^\omega = \{\sigma | \sigma : \mathbb{N} \to \mathbb{N}\}$$

To define the accessor function **head** and destructor function **tail** alluded to above, we proceed as follows:

$$\mathbf{head} \quad : \quad \mathbb{N}^\omega \to \mathbb{N} \qquad \mathbf{tail} : \mathbb{N}^\omega \to \mathbb{N}^\omega \tag{1}$$

$$\mathbf{head}(\sigma) \quad = \quad \sigma(0) \qquad \mathbf{tail}(\sigma) = (\sigma(1), \sigma(2), \ldots) \tag{2}$$

Another standard example that is often used to illustrate coalgebras, and provides a foundation for many AI and ML applications, is that of a *labelled transition system*.

Definition 8. A **labelled transition system** (LTS) $(S, \to_S, A)$ is defined by a set S of states, a transition relation $\to_S \subseteq S \times A \times S$, and a set A of labels (or equivalently, "inputs" or "actions"). We can define the transition from state s to s' under input a by the transition diagram $s \xrightarrow{a} s'$, which is equivalent to writing $\langle s, a, s' \rangle \in \to_S$. The $\mathcal{F}$-coalgebra for an LTS is defined by the functor

$$\mathcal{F}(X) = \mathcal{P}(A \times X) = \{V | V \subseteq A \times X\}$$

Just as before, we can also define a category of F-coalgebras over any category $\mathcal{C}$, where each object is a coalgebra, and the morphism between two coalgebras is defined as follows, where $f : A \to B$ is any morphism in the category $\mathcal{C}$.

Definition 9. Let $F : \mathcal{C} \to \mathcal{C}$ be an endofunctor. A *homomorphism* of F-coalgebras (A, α) and (B, β) is an arrow $f : A \to B$ in the category $\mathcal{C}$ such that the following diagram commutes:

$$\begin{array}{ccc} A & \xrightarrow{f} & B \\ \downarrow{\scriptstyle \alpha} & & \downarrow{\scriptstyle \beta} \\ F(A) & \xrightarrow{F(f)} & F(B) \end{array}$$

For example, consider two labelled transition systems $(S, A, \to_S)$ and $(T, A, \to_T)$ over the same input set A, which are defined by the coalgebras (S, α_S) and (T, α_T), respectively. An F-homomorphism $f : (S, \alpha_S) \to (T, \alpha_T)$ is a function $f : S \to T$ such that $F(f) \circ \alpha_S = \alpha_T \circ f$. Intuitively, the meaning of a homomorphism between two labeled transition systems means that:

- For all $s' \in S$, for any transition $s \xrightarrow{a}_S s'$ in the first system (S, α_S), there must be a corresponding transition in the second system $f(s) \xrightarrow{a}_T f(s;)$ in the second system.
- Conversely, for all $t \in T$, for any transition $t \xrightarrow{a}_T t'$ in the second system, there exists two states $s, s' \in S$ such that $f(s) = t, f(t) = t'$ such that $s \xrightarrow{a}_S s'$ in the first system.

If we have an F-homomorphism $f : S \to T$ with an inverse $f^{-1} : T \to S$ that is also a F-homomorphism, then the two systems $S \simeq T$ are isomorphic. If the mapping f is *injective*, we have a *monomorphism*. Finally, if the mapping f is a surjection, we have an *epimorphism*.

The analog of congruence in universal algebras is *bisimulation* in universal coalgebras. Intuitively, bisimulation allows us to construct a more "abstract" representation of a dynamical system that is still faithful to the original system. We will explore many applications of the concept of bisimulation to AI and ML systems in this paper. We introduce the concept in its general setting first, and then in the next section, we will delve into concrete examples of bisimulations.

Definition 10. Let (S, α_S) and (T, α_T) be two systems specified as coalgebras acting on the same category $\mathcal{C}$. Formally, a F-**bisimulation** for coalgebras defined on a set-valued functor $F : \mathbf{Set} \to \mathbf{Set}$ is a relation $R \subset S \times T$ of the Cartesian product of S and T is a mapping $\alpha_R : R \to F(R)$ such that the projections of R to S and T form valid F-homomorphisms.

$$\begin{array}{ccc} R & \xrightarrow{\pi_1} & S \\ \downarrow{\scriptstyle \alpha_R} & & \downarrow{\scriptstyle \alpha_S} \\ F(R) & \xrightarrow{F(\pi_1)} & F(S) \end{array}$$

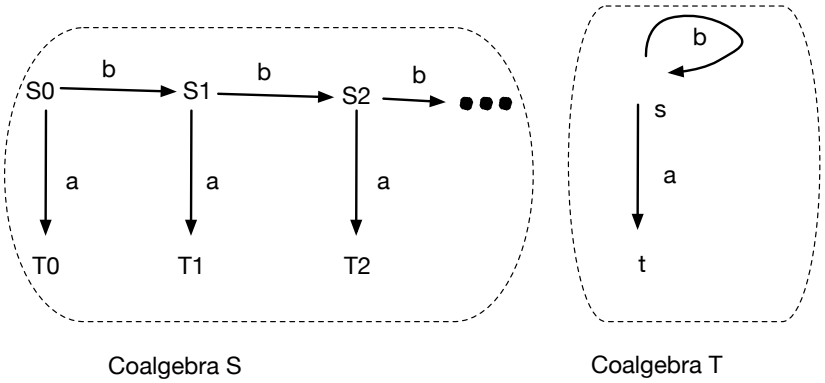


Figure 15: A bisimulation among two coalgebras.

$$\begin{array}{ccc} R & \xrightarrow{\pi_2} & T \\ \downarrow{\scriptstyle \alpha_R} & & \downarrow{\scriptstyle \alpha_T} \\ F(R) & \xrightarrow{F(\pi_2)} & F(T) \end{array}$$

Here, π_1 and π_2 are projections of the relation R onto S and T, respectively. Note the relationships in the two commutative diagrams should hold simultaneously, so that we get

$$\begin{aligned} F(\pi_1) \circ \alpha_R &= \alpha_S \circ \pi_1 \\ F(\pi_2) \circ \alpha_R &= \alpha_T \circ \pi_2 \end{aligned}$$

Intuitively, these properties imply that we can "run" the joint system R for one step, and then project onto the component systems, which gives us the same effect as if we first project the joint system onto each component system, and then run the component systems. More concretely, for two labeled transition systems that were considered above as an example of an F-homomorphism, an F-bisimulation between (S, α_S) and (T, α_T) means that there exists a relation $R \subset S \times T$ that satisfies for all $\langle s, t \rangle \in R$

- For all $s' \in S$, for any transition $s \xrightarrow{a}_S s'$ in the first system (S, α_S), there must be a corresponding transition in the second system $f(s) \xrightarrow{a}_T f(s;)$ in the second system, so that $\langle s', t' \rangle \in R$
- Conversely, for all $t \in T$, for any transition $t \xrightarrow{a}_T t'$ in the second system, there exists two states $s, s' \in S$ such that $f(s) = t, f(t) = t'$ such that $s \xrightarrow{a}_S s'$ in the first system, and $\langle s', t' \rangle \in R$.

A simple example of a bisimulation of two coalgebras is shown in Figure 15.

There are a number of basic properties about bisimulations, which we will not prove, but are useful to summarize here:

- If (R, α_R) is a bisimulation between systems S and T, the inverse R^{-1} of R is a bisimulation between systems T and S.
- Two homomorphisms $f : T \to S$ and $g : T \to U$ with a common domain T define a *span*. The *image* of the span $\langle f, g \rangle(T) = \{\langle f(t), g(t) \rangle | t \in T\}$ of f and g is also a bisimulation between S and U.
- The composition $R \circ Q$ of two bisimulations $R \subseteq S \times T$ and $Q \subseteq T \times U$ is a bisimulation between S and U.
- The union $\cup_k R_k$ of a family of bisimulations between S and T is also a bisimulation.
- The set of all bisimulations between systems S and T is a complete lattice, with least upper bounds and greatest lower bounds given by:

$$\bigvee_k R_k = \bigcup_k R_k$$

$$\bigwedge_K R_k = \bigcup \{R | R \text{ is a bisimulation between} S \text{ and } T \text{ and } R \subseteq \cap_k R_k\}$$

- The kernel $K(f) = \{\langle s, s' \rangle | f(s) = f(s')\}$ of a homomorphism $f : S \to T$ is a bisimulation equivalence.

3.2 Backpropagation as a Coalgebra

Finally, we return to the original goal of this section, which is to argue that any generative AI machine learning method can be usefully modeled not just as a functor, but rather as an endofunctor that maps an object in a category `Param` of parameters into a new object as a result of doing a machine learning step, such as a gradient update. We can now formally define backpropagation as a coalgebra over the caetgory `Param` as follows.

Recall that the category `Param` defines a strict symmetric monoidal category whose objects are Euclidean spaces, and whose morphisms $f : \mathbb{R}^n \to \mathbb{R}^m$ are equivalence classes of differential parameterized functions. To see why the backpropagation algorithm can be defined as an endofunctor over the symmetric monoidal category `Param`, recall from the previous section that backpropagation was viewed as a functor from the cateory `Param` to the cateogry `Learn`.

$$L_{\epsilon,e} : \texttt{Param} \to \texttt{Learn}$$

where $\epsilon > 0$ is a real number defining the learning rate for backpropagation, and $e(x, y) : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ is a differentiable error function such that $\frac{\partial e}{\partial x}(x_0, -)$ is invertible for each $x_0 \in \mathbb{R}$. This functor essentially defines an update procedure for each parameter in a compositional learner. In other words, the functor $L_{\epsilon,e}$ defined by backpropagation sends each parameterized function $I : P \times A \to B$ to the learner (P, I, U_I, r_I)

$$U_I(p, a, b) \coloneqq p - \epsilon \nabla_p E_I(p, a, b)$$

$$r_I(p, a, b) \coloneqq f_a(\nabla_a E_I(p, a, b))$$

where $E_I(p, a, b) \coloneqq \sum_j e(I_j(p, a), b_j)$ and f_a is a component-wise application of the inverse to $\frac{\partial e}{\partial x}(a_i, -)$ for each i.

But a simpler and we argue more elegant characterization of backpropagation is to view it as a coalgebra or dynamical system defined by an endofunctor on `Param`. Here, we view the inputs A and outputs B as the input "symbols" and output produced by a dynamical system. The actual process of updating the parameters need not be defined as "gradient descent", but it can involve any other functor (as we saw earlier, it could involve a stochastic approximation method Borkar [2008]). Our revised definition of backpropagation as an endofunctor follows. Note that this definition is generic, and applies to virtually any approach to building foundation models that updates each object to a new object in the category `Param` as a result of processing a data instance.

Definition 11. Backpropagation defines an F_B-coalgebra over the symmetric monoidal category `Param`, specified by an endofunctor $X \to F_B(X)$ defined as

$$F_B(X) = A \times B \times X$$

Note that in this definition, the endofunctor F_B takes an object X of `Param`, which is a set of network weights of a generative AI model, and produces a new set of weights, where A is the "input" symbol of the dynamical system and B is the output symbol.

3.3 Zeroth-Order Deep Learning using Stochastic Approximation

To illustrate how the broader coalgebraic definition of backpropagation is more useful than the previous definition in Fong et al. [2019], we describe a class of generative AI methods based on adapting stochastic approximation Robbins and Monro [1951] to deep learning, which are popularly referred to zeroth-order optimization Liu et al. [2009] (see Figure 16). A vast range of stochastic approximation methods have been explored in the literature (e.g., see Borkar [2008], Kushner and Yin [2003]). For example, in *random directions* stochastic approximation, each parameter is adjusted in a random direction by sampling from distribution, such as a multivariate normal, or a uniform distribution. Any of these zeroth-order stochastic approximation algorithms can itself be defined as a functor over symmetric monoidal categories

$$L^0_\epsilon : \texttt{Param} \to \texttt{Learn}$$

where $\epsilon > 0$ is a real number defining a learning rate parameter that is gradually decayed. Notice now that the error of the approximation with respect to the target plays no role in the update process itself. backpropagation, and $e(x, y) : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ is a differentiable error function such that $\frac{\partial e}{\partial x}(x_0, -)$ is invertible for each $x_0 \in \mathbb{R}$. The functor L^0_ϵ defined by zeroth-order optimization sends each parameterized function $I : P \times A \to B$ to the learner (P, I, U^0_I, r^0_I)

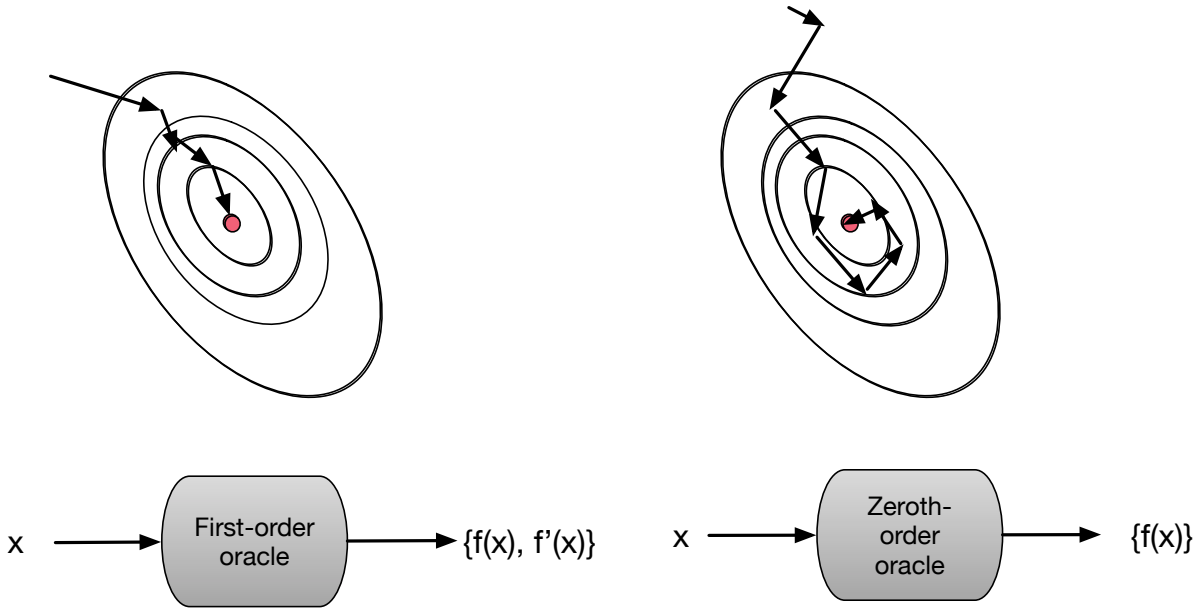


Figure 16: Zeroth-order optimization methods for generative AI are based on stochastic approximation, and average noisy values of the function to approximate gradient steps. Such methods define probabilistic coalgebras Sokolova [2011].

$$U_I^0(p, a, b) \coloneqq p - \epsilon I(p, a, b)$$

Here, the 1-point gradient estimate is approximated by the (noisy) sampled value, averaged over multiple steps using a decaying learning rate as required by the convergence theorems of stochastic approximation Robbins and Monro [1951], Kushner and Yin [2003]. The advantages of zeroth-order stochastic approximation methods are that it avoids computing gradients over a very large number of parameters (which for state of the art generative AI models can be in the billions or trillions of parameters), and it potentially helps avoid local minima by stochastically moving around in a very high-dimensional space. The disadvantage is that it can be significantly slower than gradient-based methods for well-behaved (convex) functions.

We can easily extend our previous definition of backpropagation as a coalgebra to capture zeroth-order optimization methods which act like stochastic dynamical systems, where there is a distribution of possible "next" states that is produced as a result of doing stochastic approximation updates.

Definition 12. Stochastic Backpropagation defines an F_{SGD}-coalgebra over the symmetric monoidal category `Param`, specified by an endofunctor $X \to F_{\text{SGD}}B(X)$ defined as

$$F_{\text{SGD}}(X) = A \times B \times \mathcal{D}(X)$$

where F_{SGD} defines the variant of backpropagation defined by stochastic gradient descent, and $\mathcal{D}$ is the distribution functor over X that defines a distribution over possible objects X in `Param`. There is a vast literature on stochastic coalgebras that can be defined in terms of such distribution functors. Sokolova [2011] contains an excellent review of this literature.

3.4 Lambek's Theorem and Final Coalgebras: Analyzing the Convergence of Generative AI Algorithms

Another advantage modeling backpropagation as a coalgebra defined by an endofunctor is that it provides an elegant way to analyze the problem of convergence of the algorithm to some (local) minimum solution. We explain the general principle of using final coalgebras as a generalization of (greatest) fixed points in this section. Later in Section 6, when we introduce the metric Yoneda Lemma, we will show how to use the metric coinduction property to analyze convergence of backpropagation.

Let us illustrate the concept of final coalgebras defined by a functor that represents a monotone function over the category defined by a preorder $(S, \leqslant)$, where S is a set and $\leqslant$ is a relation that is reflexive and transitive. That is,

$a \leqslant a, \forall a \in S$, and if $a \leqslant b$, and $b \leqslant c$, for $a, b, c \in S$, then $a \leqslant c$. Note that we can consider $(S, \leqslant)$ as a category, where the objects are defined as the elements of S and if $a \leqslant b$, then there is a unique arrow $a \to b$.

Let us define a functor F on a preordered set $(S, \leqslant)$ as any monotone mapping $F : S \to S$, so that if $a \leqslant b$, then $F(a) \leqslant F(b)$. Now, we can define an F-algebra as any *pre-fixed point* $x \in S$ such that $F(x) \leqslant x$. Similarly, we can define any *post-fixed point* to be any $x \in S$ such that $x \leqslant F(x)$. Finally, we can define the *final F-coalgebra* to be the *greatest post-fixed point* $x \leqslant F(x)$, and analogously, the *initial F-algebra* to be the least pre-fixed point of F.

In this section, we give a detailed overview of the concept of *final coalgebras* in the category of coalgebras parameterized by some endofunctor F. This fundamental notion plays a central role in the application of universal coalgebras to model a diverse range of AI and ML systems. Final coalgebras generalize the concept of (greatest) fixed points in many areas of application in AI, including causal inference, game theory and network economics, optimization, and reinforcement learning among others. The final coalgebra, simply put, is just the final object in the category of coalgebras. From the universal property of final objects, it follows that for any other object in the category, there must be a unique morphism to the final object. This simple property has significant consequences in applications of the coalgebraic formalism to AI and ML, as we will see throughout this paper.

An F-system (P, π) is termed **final** if for another F-system (S, α_S), there exists a unique homomorphism $f_S : (S, \alpha_S) \to (P, \pi)$. That is, (P, π) is the terminal object in the category of coalgebras Set_F defined by some set-valued endofunctor F. Since the terminal object in a category is unique up to isomorphism, any two final systems must be isomorphic.

Definition 13. An F-coalgebra (A, α) is a *fixed point* for F, written as $A \simeq F(A)$ if α is an isomorphism between A and $F(A)$. That is, not only does there exist an arrow $A \to F(A)$ by virtue of the coalgebra α, but there also exists its inverse $\alpha^{-1} : F(A) \to A$ such that

$$\alpha \circ \alpha^{-1} = \mathbf{id}_{F(A)} \text{ and } \alpha^{-1} \circ \alpha = \mathbf{id}_A$$

The following lemma was shown by Lambek, and implies that the transition structure of a final coalgebra is an isomorphism.

Theorem 1. Lambek: A final F-coalgebra is a fixed point of the endofunctor F.

Proof: The proof is worth including in this paper, as it provides a classic example of the power of diagram chasing. Let (A, α) be a final F-coalgebra. Since $(F(A), F(\alpha)$ is also an F-coalgebra, there exists a unique morphism $f : F(A) \to A$ such that the following diagram commutes:

$$\begin{array}{ccc} F(A) & \xrightarrow{f} & A \\ \downarrow{\scriptstyle F(\alpha)} & & \downarrow{\scriptstyle \alpha} \\ F(F(A)) & \xrightarrow{F(f)} & F(A) \end{array}$$

However, by the property of finality, the only arrow from (A, α) into itself is the identity. We know the following diagram also commutes, by virtue of the definition of coalgebra homomorphism:

$$\begin{array}{ccc} A & \xrightarrow{\alpha} & F(A) \\ \downarrow{\scriptstyle \alpha} & & \downarrow{\scriptstyle \alpha} \\ F(A) & \xrightarrow{F(\alpha)} & F(F(A)) \end{array}$$

Combining the above two diagrams, it clearly follows that $f \circ \alpha$ is the identity on object A, and it also follows that $F(\alpha) \circ F(f)$ is the identity on $F(A)$. Therefore, it follows that:

$$\alpha \circ f = F(f) \circ F(\alpha) = F(f \circ \alpha) = F(\mathbf{id}_A) = \mathbf{id}_{F(A)} \quad \square$$

By reversing all the arrows in the above two commutative diagrams, we get the easy duality that the initial object in an F-algebra is also a fixed point.

Theorem 2. Dual to Lambek: The initial F-algebra (A, α), where $\alpha : F(A) \to A$, in the category of F-algebras is a fixed point of F.

The proof of the duality follows in the same way, based on the universal property that there is a unique morphism from the initial object in a category to any other object.

Lambek's lemma has many implications, one of which is that final coalgebras generalize the concept of a (greatest) fixed point, which can be applied to analyze the convergence of generative AI methods, such as backpropagation. Generally speaking, in optimization, we are looking to find a solution $x \in X$ in some space X that minimizes a smooth real-valued function $f : X \to \mathbb{R}$. Since f is smooth, a natural algorithm is to find its minimum by computing the gradient ∇f over the space X. The function achieves its minimum if $\nabla f = 0$, which can be written down as a fixed point equation.

3.5 Metric Coinduction for Generative AI

To make the somewhat abstract discussion of final coalgebras above a bit more concrete, we now briefly describe the concept of *metric coinduction* Kozen and Ruozzi [2009], which is a special case of the general principle of coinduction Aczel [1988], Rutten [2000]. The basic idea is simple to describe, and is based on viewing algorithms as forming contraction mappings in a metric space. The novelty here for many readers is understanding how this well-studied notion of contractive mappings is related to coinduction and coalgebras. One way to analyze the convergence of iterative methods like backproapgation is to see if they can be shown to form contraction mappings in a metric space.

Consider a complete metric space (V, d) where $d : V \times V \to (0, 1)$ is a symmetric distance function that satisfies the triangle inequality, and all Cauchy sequences in V converge (We will see later that the property of completeness itself follows from the Yoneda Lemma!). A function $H : V \to V$ is *contractive* if there exists $0 \leqslant c < 1$ such that for all $u, v \in V$,

$$d(H(u), H(v)) \leqslant c \cdot d(u, v)$$

In effect, applying the algorithm represented by the continuous mapping H causes the distances between u and v to shrink, and repeated application eventually guarantees convergence to a fixed point. The novelty here is to interpret the fixed point as a final coalgebra. We will later see that the concept of a (greatest) fixed point is generalized by the concept of final coalgebras.

Definition 14. Metric Coinduction Principle Kozen and Ruozzi [2009]: If ϕ is a closed nonempty subset of a complete metric space V, and if H is an eventually contractive map on V that preserves ϕ, then the unique fixed point u^* of H is in ϕ. In other words, we can write:

$$\exists u \phi(u), \quad \forall \phi(u) \quad \Rightarrow \quad \begin{matrix} \phi(H(u)) \\ \phi(u^*) \end{matrix}$$

It should not be surprising to those familiar with contraction mapping style arguments that a large number of applications in AI and ML, including game theory, reinforcement learning and stochastic approximation involve proofs of convergence that exploit properties of contraction mappings. What might be less familiar is how to think of this in terms of the concept of *coinductive inference*. To explain this perspective briefly, let us introduce the relevant category theoretic terminology.

Let us define a category C whose objects are nonempty closed subsets of V, and whose arrows are reverse set inclusions. That is, there is a unique arrow $\phi_1 \to \phi_2$ if $\phi_1 \supseteq \phi_2$. Then, we can define an *endofunctor* $\bar{H}$ as the closure mapping $\bar{H}(\phi) = \text{cl}(H(\phi))$, where cl denotes the closure in the metric topology of V. Note that $\bar{H}$ is an endofunctor on C because $\bar{H}(\phi_1) \supseteq \bar{H}(\phi_2)$ whenever $\phi_1 \supseteq \phi_2$.

Definition 15. An $\bar{H}$**-coalgebra** is defined as the pair $(\phi, \phi \supseteq \bar{H}(\phi))$ (or equivalently, we can write $\phi \subseteq H(\phi)$. The **final coalgebra** is the isomorphism $u^* \simeq \bar{H}(u^*)$ where u^* is the unique fixed point of the mapping H. The metric coinduction rule can be restated more formally in this case as:

$$\phi \supseteq H(\phi) \;\; \Rightarrow \;\; \phi \supseteq H(u^*)$$

This result has deep significance, as we will see later, and provides an elegant way to prove contraction style arguments in very general settings. In particular, it can be applied to analyze the convergence of machine learning methods for GAIA models to see if they can be shown to convergence in a (generalized) metric space. We will postpone the details of this analysis to a subsequent paper

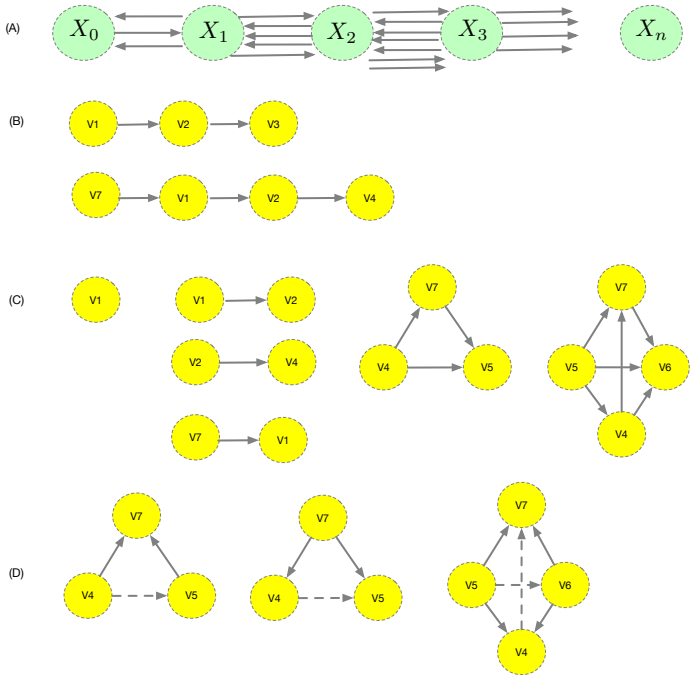


Figure 17: (A) GAIA is based on a *hierarchical* simplicial sets and objects. The base simplicial set X_0 is a set of entities that can be mapped to computational entities in generative AI, such as a tokens in a large language model, or images in a diffusion based system. The set X_1 defines a collection of morphisms between pairs of objects in X_0, where each morphism could define a deep learning module as explained in Section 2. (B) The first two sets X_0 and X_1 essentially define what is possible with today's compositionally based generative AI system using backpropagation, where learning is conceived of as an entirely sequential process. (C) X_2 and higher-level simplicial sets constitute the novel core of GAIA: here, groups of sub-simplicial objects act like business units with a common objective. Each n-simplex has $n+1$ sub-simplicial complexes, and information is transmitted hierarchically in GAIA from superior simplicial sets to subordinate simplicial sets using lifting diagrams. (D) Solving "outer horn" extension problems is more challenging for methods like deep learning with backpropagation, than solving "inner horn" extensions.

4 Layer 1 of GAIA: Simplicial Sets for Generative AI

Unlike earlier generative AI architectures, GAIA uses the paradigm of simplicial sets and objects as the basic building blocks for generative AI (see Figure 3). We can still cast much of the earlier work described above in terms of GAIA, but this flexibility gives us the power to also formulate generative AI approaches that lie beyond the scope of compositional learning methods, such as backpropagation. As we illustrated in Figure 6, GAIA puts together building blocks of generative AI methods as n-simplices of a simplicial set. In the simplest setting, these combine compositionally in the way that Fong et al. [2019] defined for the category `Learn` that we discussed in detail in the previous section. It is possible to take the category `Learn` and embed it in the category of simplicial sets using the *nerve functor* Lurie [2022], which is a full and faithful embedding of the category as a simplicial set. Each n-simplex is then defined by n-length sequences of a generative AI system, like a Transformer building block that computes a permutation-equivariant map. But the left adjoint of the nerve functor that maps back from the simplicial set category is a "lossy" functor that does not generate a full and faithful embedding, which shows why simplicial learning is more powerful in principle than compositional learning.

The first layer of GAIA is based on the simplicial category Δ, which serves as a "combinatorial factory" for piecing together building blocks of generative AI systems into larger units, and for decomposing complex systems into their component subsystems. The category Δ is defined over ordinal numbers $[n], n \geqslant 0$, but really "comes to life" when it is plugged into some concrete category, such as the ones described in the previous section like `Learn` or `Para`. For example, if the parameters of a learning method are defined over a category of **Sets**, then a contravariant functor from Δ into sets is called a simplicial set May [1992]. We can also define functors from Δ into some category of generative AI models, like Transformers Vaswani et al. [2017] or structured state space sequence (S4) models Gu et al. [2022] or diffusion models Song and Ermon [2019].

4.1 Simplicial Sets and Objects

As shown in Figure 17, a simplicial set can be viewed as a collection of sets, or a *graded* set, $S_n, n \geqslant 0$, where S_0 defines the primitive objects (which can be elements of the category `Param` defined in the previous section), S_1 represents a collection of "edge" objects (which can be viewed as Learners as defined in the previous section), S_2 represents simplices of three objects interacting, and in general, S_n defines a collection of objects that represents interactions of order n. These higher-level simplicial sets act like "business units" in a company: they have a hierarchical structure, receive inputs and outputs from higher-level superiors and lower-level subordinates, and adjust their internal parameters. These n-simplicial sets are related to each other by *degeneracy* operators that map S_n into S_{n+1} or *face* operators that map S_n into S_{n-1}. Figure 6 shows an example of a 3-simplex. Note how in Figure 3, the simplicial set X_3 sends "back" information to X_2 through four face operators. These exactly correspond to the four subsimplices of each object in X_3, as illustrated in Figure 3, because each 3-simplex has four faces. The crux of the GAIA framework is to treat each such simplex as a building block of a generative AI system.

Simplicial sets are higher-dimensional generalizations of directed graphs, partially ordered sets, as well as regular categories themselves. Importantly, simplicial sets and simplicial objects form a foundation for higher-order category theory. Simplicial objects have long been a foundation for algebraic topology, and more recently in higher-order category theory. The category Δ has non-empty ordinals $[n] = \{0, 1, \ldots, n]$ as objects, and order-preserving maps $[m] \to [n]$ as arrows. An important property in Δ is that any many-to-many mapping is decomposable as a composition of an injective and a surjective mapping, each of which is decomposable into a sequence of elementary injections $\delta_i : [n] \to [n+1]$, called *coface* mappings, which omits $i \in [n]$, and a sequence of elementary surjections $\sigma_i : [n] \to [n-1]$, called *co-degeneracy* mappings, which repeats $i \in [n]$. The fundamental simplex $\Delta([n])$ is the presheaf of all morphisms into $[n]$, that is, the representable functor $\Delta(-, [n])$. The Yoneda Lemma assures us that an n-simplex $x \in X_n$ can be identified with the corresponding map $\Delta[n] \to X$. Every morphism $f : [n] \to [m]$ in Δ is functorially mapped to the map $\Delta[m] \to \Delta[n]$ in $\mathcal{S}$.

Any morphism in the category Δ can be defined as a sequence of *co-degeneracy* and *co-face* operators, where the co-face operator $\delta_i : [n-1] \to [n], 0 \leqslant i \leqslant n$ is defined as:

$$\delta_i(j) = \begin{cases} j, & \text{for } 0 \leqslant j \leqslant i-1 \\ j+1 & \text{for } i \leqslant j \leqslant n-1 \end{cases}$$

Analogously, the co-degeneracy operator $\sigma_j : [n+1] \to [n]$ is defined as

$$\sigma_j(k) = \begin{cases} j, & \text{for } 0 \leqslant k \leqslant j \\ k-1 & \text{for } j < k \leqslant n+1 \end{cases}$$

Note that under the contravariant mappings, co-face mappings turn into face mappings, and co-degeneracy mappings turn into degeneracy mappings. That is, for any simplicial object (or set) X_n, we have $X(\delta_i) \coloneqq d_i : X_n \to X_{n-1}$, and likewise, $X(\sigma_j) \coloneqq s_j : X_{n-1} \to X_n$.

The compositions of these arrows define certain well-known properties May [1992], Richter [2020]:

$$\begin{aligned} \delta_j \circ \delta_i &= \delta_i \circ \delta_{j-1}, \;\; i < j \\ \sigma_j \circ \sigma_i &= \sigma_i \circ \sigma_{j+1}, \;\; i \leqslant j \\ \sigma_j \circ \delta_i(j) &= \begin{cases} \sigma_i \circ \sigma_{j+1}, & \text{for } i < j \\ 1_{[n]} & \text{for } i = j, j+1 \\ \sigma_{i-1} \circ \sigma_j, \text{for } i > j+1 \end{cases} \end{aligned}$$

Example 1. The "vertices" of a simplicial object $\mathcal{C}_n$ are the objects in $\mathcal{C}$, and the "edges" of $\mathcal{C}$ are its arrows $f : X \to Y$, where X and Y are objects in $\mathcal{C}$. Given any such arrow, the degeneracy operators $d_0 f = Y$ and $d_1 f = X$ recover the source and target of each arrow. Also, given an object X of category $\mathcal{C}$, we can regard the face operator $s_0 X$ as its identity morphism $\mathbf{1}_X : X \to X$.

Example 2. Given a category $\mathcal{C}$, we can identify an n-simplex σ of a simplicial set $\mathcal{C}_n$ with the sequence:

$$\sigma = C_o \xrightarrow{f_1} C_1 \xrightarrow{f_2} \ldots \xrightarrow{f_n} C_n$$

the face operator d_0 applied to σ yields the sequence

$$d_0\sigma = C_1 \xrightarrow{f_2} C_2 \xrightarrow{f_3} \ldots \xrightarrow{f_n} C_n$$

where the object C_0 is "deleted" along with the morphism f_0 leaving it.

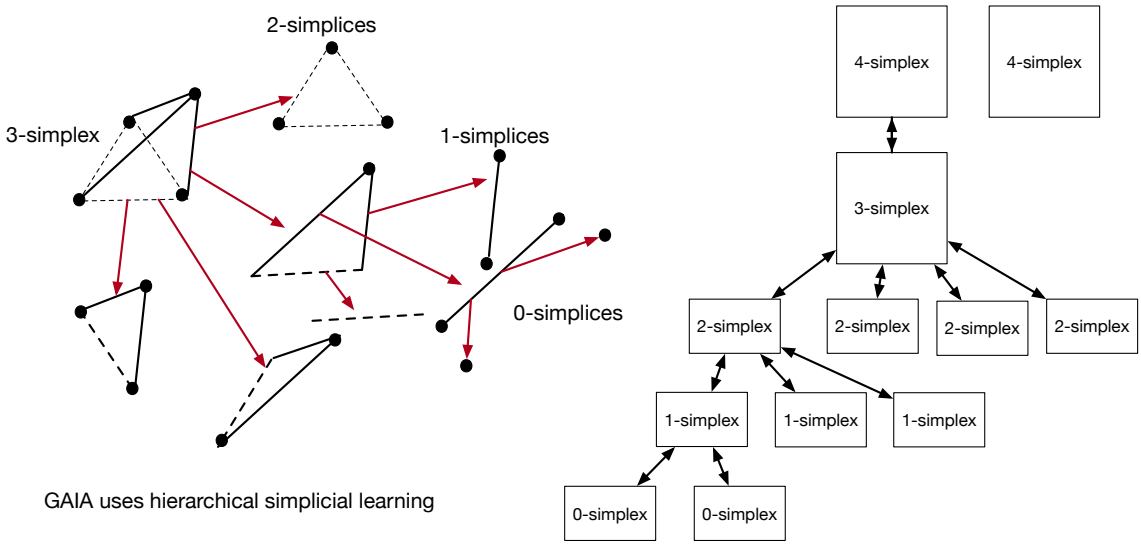


Figure 18: Example of a "small business unit" in GAIA defined as a 3-simplex that maintains its set of internal parameters, and updates them based on information it receives from its superiors and subordinates.

Example 3. Given a category $\mathcal{C}$, and an n-simplex σ of the simplicial set $\mathcal{C}_n$, the face operator d_n applied to σ yields the sequence

$$d_n\sigma = C_0 \xrightarrow{f_1} C_1 \xrightarrow{f_2} \dots \xrightarrow{f_{n-1}} C_{n-1}$$

where the object C_n is "deleted" along with the morphism f_n entering it. Note this face operator can be viewed as analogous to interventions on leaf nodes in a causal DAG model.

Example 4. Given a category $\mathcal{C}$, and an n-simplex σ of the simplicial set $\mathcal{C}_n$ the face operator $d_i, 0 < i < n$ applied to σ yields the sequence

$$d_i\sigma = C_0 \xrightarrow{f_1} C_1 \xrightarrow{f_2} \dots C_{i-1} \xrightarrow{f_{i+1}\circ f_i} C_{i+1} \dots \xrightarrow{f_n} C_n$$

where the object C_i is "deleted" and the morphisms f_i is composed with morphism f_{i+1}. Note that this process can be abstractly viewed as intervening on object C_i by choosing a specific value for it (which essentially "freezes" the morphism f_i entering object C_i to a constant value).

Example 5. Given a category $\mathcal{C}$, and an n-simplex σ of the simplicial set $\mathcal{C}_n$, the degeneracy operator $s_i, 0 \leqslant i \leqslant n$ applied to σ yields the sequence

$$s_i\sigma = C_0 \xrightarrow{f_1} C_1 \xrightarrow{f_2} \dots C_i \xrightarrow{\mathbf{1}_{C_i}} C_i \xrightarrow{f_{i+1}} C_{i+1} \dots \xrightarrow{f_n} C_n$$

where the object C_i is "repeated" by inserting its identity morphism $\mathbf{1}_{C_i}$.

Definition 16. Given a category $\mathcal{C}$, and an n-simplex σ of the simplicial set $\mathcal{C}_n$, σ is a **degenerate** simplex if some f_i in σ is an identity morphism, in which case C_i and C_{i+1} are equal.

4.2 Hierarchical Learning in GAIA by solving Lifting Problems

As we mentioned earlier, a crucial difference between GAIA and earlier generative AI architectures is that it is based on a hierarchical model of simplicial learning, rather than the standard compositional learning framework described in Section 2. To understand how such a structure will work, we need to define some key ideas from higher-order category theory below, but before we do that, we want to build up some intuition as to how this process will work at a more informal level. Figure 18 illustrates the idea using the same figure we showed earlier as Figure 3, but here, we will use it to illustrate the simplicial learning concept.

To understand how simplicial learning works, let us consider as an example the 3-simplex shown in Figure 18. We generalize the earlier compositional model defined in Section 4, where in the category `Learn` (see Figure 10), each learner was a morphism in the symmetric monoidal category. In that model, each learner morphism transmits information downstream to its successors and upstream to its predecessors, but there is no hierarchical structure. Here, in Figure 18, the simplicial structure defines a hierarchy of learners, so that each learner is not just a morphism anymore, but a n-simplex that maintains its internal set of parameters that it then updates based on the information from its superiors

and subordinates. To define this more carefully, we can construct a functor that maps the algebraic structure of a simplicial set Δ into a suitable parameter space (e.g., a symmetric monoidal category like vector spaces), whereby each n-simplex now becomes defined as a contravariant functor $\Delta^{op} \rightarrow \mathbf{Vect}$.

We can in fact use exactly the same updates defined earlier in Section 4, following Fong et al. [2019], with the crucial difference being that the updates must be consistent across the hierarchical structure of the simplicial complex. So, each n-simplex is updated based on data from its subordinate $n-1$ sub-simplicial complexes and its superior $n+1$-simplicial complexes, but these need to be made consistent with each other. To solve this problem requires some additional machinery from higher-order category theory, which we now introduce below.

Lifting problems provide elegant ways to define solutions to computational problems in category theory regarding the existence of mappings. We will use these lifting diagrams later in this paper. For example, the notion of injective and surjective functions, the notion of separation in topology, and many other basic constructs can be formulated as solutions to lifting problems. Lifting problems define ways of decomposing structures into simpler pieces, and putting them back together again.

Definition 17. Let $\mathcal{C}$ be a category. A **lifting problem** in $\mathcal{C}$ is a commutative diagram σ in $\mathcal{C}$.

$$\begin{array}{ccc} A & \xrightarrow{\mu} & X \\ \downarrow{\scriptstyle f} & & \downarrow{\scriptstyle p} \\ B & \xrightarrow{\nu} & Y \end{array}$$

Definition 18. Let $\mathcal{C}$ be a category. A **solution to a lifting problem** in $\mathcal{C}$ is a morphism $h : B \rightarrow X$ in $\mathcal{C}$ satisfying $p \circ h = \nu$ and $h \circ f = \mu$ as indicated in the diagram below.

$$\begin{array}{ccc} A & \xrightarrow{\mu} & X \\ \downarrow{\scriptstyle f} & \overset{h}{\nearrow} & \downarrow{\scriptstyle p} \\ B & \xrightarrow{\nu} & Y \end{array}$$

Definition 19. Let $\mathcal{C}$ be a category. If we are given two morphisms $f : A \rightarrow B$ and $p : X \rightarrow Y$ in $\mathcal{C}$, we say that f has the **left lifting property** with respect to p, or that p has the **right lifting property** with respect to f if for every pair of morphisms $\mu : A \rightarrow X$ and $\nu : B \rightarrow Y$ satisfying the equations $p \circ \mu = \nu \circ f$, the associated lifting problem indicated in the diagram below.

$$\begin{array}{ccc} A & \xrightarrow{\mu} & X \\ \downarrow{\scriptstyle f} & \overset{h}{\nearrow} & \downarrow{\scriptstyle p} \\ B & \xrightarrow{\nu} & Y \end{array}$$

admits a solution given by the map $h : B \rightarrow X$ satisfying $p \circ h = \nu$ and $h \circ f = \mu$.

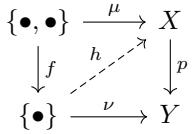
Gavrilovich [2017] shows that a remarkable number of properties in mathematics can be defined as lifting problems. Spivak [2013] showed that query answering in languages like SQL in relational databases can be formalized as lifting problems. Mahadevan [2023] showed that causal inference could be posed in terms of lifting problems. At its heart, a lifting problem defines a constrained search over a space of parameters, and it is that property that makes it so useful in generative AI because in effect, methods like backpropagation can be viewed as solving lifting problems. As a simple example to build intuition, here is a way any surjective (onto) function as a solution to a lifting problem.

Example 6. Given the paradigmatic non-surjective morphism $f : \emptyset \rightarrow \{\bullet\}$, any morphism p that has the right lifting property with respect to f is a **surjective mapping**. .

$$\begin{array}{ccc} \emptyset & \xrightarrow{\mu} & X \\ \downarrow{\scriptstyle f} & \overset{h}{\nearrow} & \downarrow{\scriptstyle p} \\ \{\bullet\} & \xrightarrow{\nu} & Y \end{array}$$

Similarly, here is another lifting problem whose solution defines an $1-1$ injective function.

Example 7. Given the paradigmatic non-injective morphism $f : \{\bullet, \bullet\} \rightarrow \{\bullet\}$, any morphism p that has the right lifting property with respect to f is an **injective mapping**. .



4.3 Simplicial Subsets and Horns in GAIA

To explain how lifting problems can be used for generative AI in GAIA, we need to define lifting problems over n-simplicial complexes. The basic idea, as illustrated in Figure 18, is that we construct a solution to a lifting problem by asking if a particular sub-simplicial complex can be "extended" into the whole complex. This extension process is essentially what methods like backpropagation are doing, and universal approximation results for Transformers Yun et al. [2020] are in effect saying that a solution to a lifting problem exists for a particular class of simplicial complexes defined as n-length sequences of Transformer models.

We first describe more complex ways of extracting parts of categorical structures using simplicial subsets and horns. These structures will play a key role in defining suitable lifting problems.

Definition 20. The **standard simplex** Δ^n is the simplicial set defined by the construction

$$([m] \in \Delta) \mapsto \mathbf{Hom}_{\Delta}([m], [n])$$

By convention, $\Delta^{-1} \coloneqq \emptyset$. The standard 0-simplex Δ^0 maps each $[n] \in \Delta^{op}$ to the single element set $\{\bullet\}$.

Definition 21. Let $S_\bullet$ denote a simplicial set. If for every integer $n \geqslant 0$, we are given a subset $T_n \subseteq S_n$, such that the face and degeneracy maps

$$d_i : S_n \to S_{n-1} \quad s_i : S_n \to S_{n+1}$$

applied to T_n result in

$$d_i : T_n \to T_{n-1} \quad s_i : T_n \to T_{n+1}$$

then the collection $\{T_n\}_{n \geqslant 0}$ defines a **simplicial subset** $T_\bullet \subseteq S_\bullet$

Definition 22. The **boundary** is a simplicial set $(\partial\Delta^n) : \Delta^{op} \to \mathbf{Set}$ defined as

$$(\partial\Delta^n)([m]) = \{\alpha \in \mathbf{Hom}_{\Delta}([m], [n]) : \alpha \text{ is not surjective}\}$$

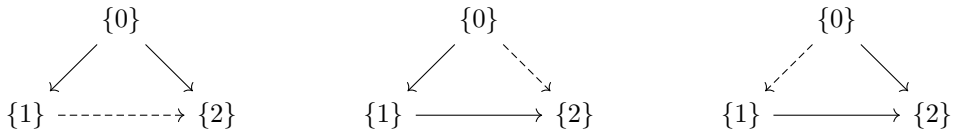
Note that the boundary $\partial\Delta^n$ is a simplicial subset of the standard n-simplex Δ^n.

Definition 23. The **Horn** $\Lambda^n_i : \Delta^{op} \to \mathbf{Set}$ is defined as

$$(\Lambda^n_i)([m]) = \{\alpha \in \mathbf{Hom}_{\Delta}([m], [n]) : [n] \not\subseteq \alpha([m]) \cup \{i\}\}$$

Intuitively, the Horn Λ^n_i can be viewed as the simplicial subset that results from removing the interior of the n-simplex Δ^n together with the face opposite its ith vertex.

Consider the problem of composing 1-dimensional simplices to form a 2-dimensional simplicial object. Each simplicial subset of an n-simplex induces a a *horn* Λ^n_k, where $0 \leqslant k \leqslant n$. Intuitively, a horn is a subset of a simplicial object that results from removing the interior of the n-simplex and the face opposite the ith vertex. Consider the three horns defined below. The dashed arrow $\dashrightarrow$ indicates edges of the 2-simplex Δ^2 not contained in the horns.



The inner horn Λ^2_1 is the middle diagram above, and admits an easy solution to the "horn filling" problem of composing the simplicial subsets. The two outer horns on either end pose a more difficult challenge. For example, filling the outer horn Λ^2_0 when the morphism between $\{0\}$ and $\{1\}$ is f and that between $\{0\}$ and $\{2\}$ is the identity $\mathbf{1}$ is tantamount to finding the left inverse of f up to homotopy. Dually, in this case, filling the outer horn Λ^2_2 is tantamount to finding the right inverse of f up to homotopy. A considerable elaboration of the theoretical machinery in category theory is required to describe the various solutions proposed, which led to different ways of defining higher-order category theory Boardman and Vogt [1973], Joyal [2002], Lurie [2009].

4.4 Higher-Order Categories

We now formally introduce higher-order categories, building on the framework proposed in a number of formalisms. We briefly summarize various approaches to the horn filling problem in higher-order category theory.

Definition 24. Let $f : X \to S$ be a morphism of simplicial sets. We say f is a **Kan fibration** if, for each $n > 0$, and each $0 \leqslant i \leqslant n$, every lifting problem.

$$\begin{array}{ccc} \Lambda_i^n & \xrightarrow{\sigma_0} & X \\ \downarrow & \overset{\sigma}{\nearrow} & \downarrow f \\ \Delta^n & \xrightarrow{\bar{\sigma}} & S \end{array}$$

admits a solution. More precisely, for every map of simplicial sets $\sigma_0 : \Lambda_i^n \to X$ and every n-simplex $\bar{\sigma} : \Delta^n \to S$ extending $f \circ \sigma_0$, we can extend σ_0 to an n-simplex $\sigma : \Delta^n \to X$ satisfying $f \circ \sigma = \bar{\sigma}$.

Example 8. Given a simplicial set X, then a projection map $X \to \Delta^0$ that is a Kan fibration is called a **Kan complex**.

Example 9. Any isomorphism between simplicial sets is a Kan fibration.

Example 10. The collection of Kan fibrations is closed under retracts.

Definition 25. Lurie [2009] An ∞-category is a simplicial object $S_\bullet$ which satisfies the following condition:

- For $0 < i < n$, every map of simplicial sets $\sigma_0 : \Lambda_i^n \to S_\bullet$ can be extended to a map $\sigma : \Delta^n \to S_i$.

This definition emerges out of a common generalization of two other conditions on a simplicial set S_i:

1. **Property K**: For $n > 0$ and $0 \leqslant i \leqslant n$, every map of simplicial sets $\sigma_0 : \Lambda_i^n \to S_\bullet$ can be extended to a map $\sigma : \Delta^n \to S_i$.
2. **Property C**: for $0 < 1 < n$, every map of simplicial sets $\sigma_0 : \Lambda_i^n \to S_i$ can be extended uniquely to a map $\sigma : \Delta^n \to S_i$.

Simplicial objects that satisfy property K were defined above to be Kan complexes. Simplicial objects that satisfy property C above can be identified with the nerve of a category, which yields a full and faithful embedding of a category in the category of sets. definition 25 generalizes both of these definitions, and was called a *quasicategory* in Joyal [2002] and *weak Kan complexes* in Boardman and Vogt [1973] when $\mathcal{C}$ is a category.

5 Layer 2 of GAIA: Generative AI using Simplicial Categories

The second layer of GAIA is based on defining generative models as universal coalgebras over some base category, including the standard mathematical categories (**Sets**, measurable spaces **Meas**, topoogical spaces **Top** or Vector spaces **Vect**). Existing approaches to generative AI, such as Transformers Vaswani et al. [2017], structured state-space models Gu et al. [2022], or image diffusion models Song and Ermon [2019] can all be defined as (stochastic) coalgebras over one of the base categories. We first introduce some basic categorical structures, and then define the category of universal coalgebras over these. to define generative AI systems as coalgebras.

5.1 Categories as Building Blocks of GAIA

GAIA is built on the hypothesis that category theory provides a universal language for encoding foundation models. We define a few salient aspects of category theory in this section, including showing how it can reveal surprising similarities between algebraic structures that superficially look very different, such as metric spaces and partially ordered sets (see Table 1). A key principle that is often exploited is to explicitly represent the structure in the collection of morphisms between two objects. That is, for some category C, the $\mathbf{Hom}_C(a, b)$ between objects a and b might itself have some additional structure beyond that of merely being a collection or a set. For example, in the category of vector spaces, the set of morphisms (linear transformations) between two vector spaces U and V is itself a vector space. So-called V-enriched categories signify cases when the **Hom** values are specified in some structure V. Examples include metric spaces, where the **Hom** values are non-negative real numbers representing distances, and partially ordered sets (posets) where the **Hom** values are Boolean.

The aim of category theory is to build a "unified field theory" of mathematics based on a simple model of *objects* that *interact* with each other, analogous to directed graph representations. In graphs, vertices represent arbitrary entities, and

Table 1: Categories are defined as collections of objects and arrows between them.

C	$\mathbf{Hom}_{\mathcal{C}}$ values	Composition and and identity law	Domain for composition	Domain for for identity laws
General category	Sets	Functions	Cartesian product	One element set
Metric spaces	Non-negative numbers	$\geqslant$	sum	zero
Posets	Truth values	Entailment	Conjunction	true
V-enriched category	objects in V	morphisms in V	tensor product in V	unit object for tensor product in V
$F : \mathcal{C}^{op} \times C \to D$ Coends, ends	Bivalent functors	Dinatural transformations	Probabilities, distances topological embeddings	Unit object

the edges denote some form of (directional) interaction. In categories, there is no restriction on how many edges exist between any two objects. In a *locally small* category, there is assumed to be a "set's worth" of edges, meaning that it could still be infinite! In addition, small categories are assumed to contain a set's worth of objects (again, which might not be finite). The framework is *compositional*, in that categories can be formed out of objects, *arrows* that define the interaction between objects, or *functors* that define the interactions between categories. This compositionality gives us a rich *generative* space of models that will be invaluable in modeling UIGs.

Category theory gives an exceptional set of "measuring tools" for modeling Transformers and other generative models in AI. Choosing a category means selecting a collection of objects and a collection of composable arrows by which each pair of objects interact. This choice of objects and arrows defines the measurement apparatus that is used in formulating and solving an imitation game. A key result called the Yoneda Lemma shows that *objects can be identified up to isomorphism solely by their interactions with other objects*. Category theory also embodies the principle of *universality*: a property is universal if it defines an *initial* or *final* object in a category. Many approaches in generative AI, such as probabilistic generative models or distance metrics, can be abstractly characterized as initial or final objects in a category of *wedges*, where the objects are bifunctors and the arrows are dinatural transformations. Loregian [2021] has an excellent treatment of the calculus of coends, which we will discuss in detail later in the paper. At a high level, the notion of object isomorphism in category theory is defined as follows.

Definition 26. Two objects X and Y in a category $\mathcal{C}$ are deemed **isomorphic**, or $X \cong Y$ if and only if there is an invertible morphism $f : X \to Y$, namely f is both *left invertible* using a morphism $g : Y \to X$ so that $g \circ f = \mathbf{id}_X$, and f is *right invertible* using a morphism h where $f \circ h = \mathbf{id}_Y$.

Category theory provides a rich language to describe how objects interact, including notions like *braiding* that plays a key role in quantum computing Coecke et al. [2016]. The notion of isomorphism can be significantly weakened to include notions like homotopy. This notion of homotopy generalizes the notion of homotopy in topology, which defines why an object like a coffee cup is topologically homotopic to a doughnut (they have the same number of "holes"). In the category **Sets**, two finite sets are considered isomorphic if they have the same number of elements, as it is then trivial to define an invertible pair of morphisms between them. In the category $\mathbf{Vect}_k$ of vector spaces over some field k, two objects (vector spaces) are isomorphic if there is a set of invertible linear transformations between them. As we will see below, the passage from a set to the "free" vector space generated by elements of the set is another manifestation of the universal arrow property. In the category of topological spaces **Top**, two objects are isomorphic if there is a pair of continuous functions that makes them *homeomorphic* May and Ponto [2012]. A more refined category is *hTop*, the category defined by topological spaces where the arrows are now given by homotopy classes of continuous functions.

Definition 27. Let C and C' be a pair of objects in a category $\mathcal{C}$. We say C is **a retract** of C' if there exists maps $i : C \to C'$ and $r : C' \to C$ such that $r \circ i = \mathrm{id}_{\mathcal{C}}$.

Definition 28. Let $\mathcal{C}$ be a category. We say a morphism $f : C \to D$ is a **retract of another morphism** $f' : C \to D$ if it is a retract of f' when viewed as an object of the functor category $\mathbf{Hom}([1], \mathcal{C})$. A collection of morphisms T of $\mathcal{C}$ is **closed under retracts** if for every pair of morphisms f, f' of $\mathcal{C}$, if f is a retract of f', and f' is in T, then f is also in T.

The point of these examples is to illustrate that choosing a category, which means choosing a collection of objects and arrows, is like defining a measurement system for deciding if two objects are isomorphic. A richer model of interaction is provided by *simplicial sets* May [1992], which is a *graded set* $S_n,\ n \geqslant 0$, where S_0 represents a set of non-interacting objects, S_1 represents a set of pairwise interactions, S_2 represents a set of three-way interactions, and so on. We can map any category into a simplicial set by constructing sequences of length n of composable morphisms. For example, we can model sequences of words in a language as composable morphisms, thereby constructing a simplicial set representation of language-based interactions in an imitation game. Then, the corresponding notion of homotopy between simplicial sets is defined as Richter [2020]:

Set theory	Category theory
set	object
subset	subobject
truth values $\{0, 1\}$	subobject classifier Ω
power set $P(A) = 2^A$	power object $P(A) = \Omega^A$
bijection	isomorphims
injection	monic arrow
surjection	epic arrow
singleton set $\{*\}$	terminal object $\mathbf{1}$
empty set $\emptyset$	initial object $\mathbf{0}$
elements of a set X	morphism $f : \mathbf{1} \to X$
-	functors, natural transformations
-	limits, colimits, adjunctions

Figure 19: Comparison of notions from set theory and category theory.

Definition 29. Let X and Y be simplicial sets, and suppose we are given a pair of morphisms $f_0, f_1 : X \to Y$. A **homotopy** from f_0 to f_1 is a morphism $h : \Delta^1 \times X \to Y$ satisfying $f_0 = h|_{0 \times X}$ and $f_1 = h_{1 \times X}$.

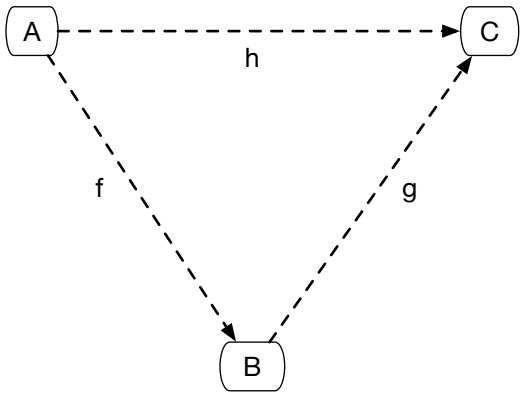


Figure 20: Category theory is a compositional model of a system in terms of objects and their interactions.

Figure 19 compares the basic notions in set theory vs. category theory. Figure 20 illustrates a simple category of 3 objects: A, B, and C that interact through the morphisms $f : A \to B$, $g : B \to C$, and $h : A \to C$. Categories involve a fundamental notion of *composition*: the morphism $h : A \to C$ can be defined as the composition $g \circ f$ of the morphisms from f and g. What the objects and morphisms represent is arbitrary, and like the canonical directed graph model, this abstractness gives category theory – like graph theory – a universal quality in terms of applicability to a wide range of problems. While categories and graphs and intimately related, in a category, there is no assumption of finiteness in terms of the cardinality of objects or morphisms. A category is defined to be *small* or *locally small* if there is a set's worth of objects and between any two objects, a set's worth of morphisms, but of course, a set need not be finite. As a simple example, the set of integers $\mathbb{Z}$ defines a category, where each integer z is an object and there is a morphism $f : a \to b$ between integers a and b if $a \leqslant b$. This example serves to immediately clarify an important point: a category is only defined if both the objects and morphisms are defined. The category of integers $\mathbb{Z}$ may be defined in many ways, depending on what the morphisms represent.

Briefly, a category is a collection of objects, and a collection of morphisms between pairs of objects, which are closed under composition, satisfy associativity, and include an identity morphism for every object. For example, sets form a category under the standard morphism of functions. Groups, modules, topological spaces and vector spaces all form categories in their own right, with suitable morphisms (e.g, for groups, we use group homomorphisms, and for vector spaces, we use linear transformations).

A simple way to understand the definition of a category is to view it as a "generalized" graph, where there is no limitation on the number of vertices, or the number of edges between any given pair of vertices. Each vertex defines an object in a category, and each edge is associated with a morphism. The underlying graph induces a "free" category where we consider all possible paths between pairs of vertices (including self-loops) as the set of morphisms between

them. In the reverse direction, given a category, we can define a "forgetful" functor that extracts the underlying graph from the category, forgetting the composition rule.

Definition 30. A **graph** $\mathcal{G}$ (sometimes referred to as a quiver) is a labeled directed multi-graph defined by a set O of *objects*, a set A of *arrows*, along with two morphisms $s : A \to O$ and $t : A \to O$ that specify the domain and co-domain of each arrow. In this graph, we define the set of composable pairs of arrows by the set

$$A \times_O A = \{\langle g, f\rangle |\ g, f \in A,\ \ s(g) = t(f)\}$$

A **category** $\mathcal{C}$ is a graph $\mathcal{G}$ with two additional functions: $\mathbf{id} : O \to A$, mapping each object $c \in C$ to an arrow $\mathbf{id}_c$ and $\circ : A \times_O A \to A$, mapping each pair of composable morphisms $\langle f, g\rangle$ to their composition $g \circ f$.

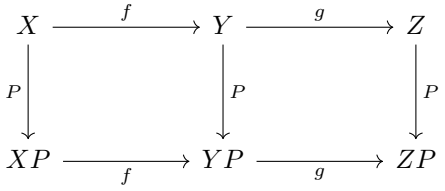
It is worth emphasizing that no assumption is made here of the finiteness of a graph, either in terms of its associated objects (vertices) or arrows (edges). Indeed, it is entirely reasonable to define categories whose graphs contain an infinite number of edges. A simple example is the group $\mathbb{Z}$ of integers under addition, which can be represented as a single object, denoted $\{\bullet\}$ and an infinite number of morphisms $f : \bullet \to \bullet$, each of which represents an integer, where composition of morphisms is defined by addition. In this example, all morphisms are invertible. In a general category with more than one object, a *groupoid* defines a category all of whose morphisms are invertible. A central principle in category theory is to avoid the use of equality, which is pervasive in mathematics, in favor of a more general notion of *isomorphism* or weaker versions of it. Many examples of categories can be given that are relevant to specific problems in AI and ML. Some examples of categories of common mathematical structures are illustrated below.

- **Set**: The canonical example of a category is **Set**, which has as its objects, sets, and morphisms are functions from one set to another. The **Set** category will play a central role in our framework, as it is fundamental to the universal representation constructed by Yoneda embeddings.
- **Top:** The category **Top** has topological spaces as its objects, and continuous functions as its morphisms. Recall that a topological space (X, Ξ) consists of a set X, and a collection of subsets Ξ of X closed under finite intersection and arbitrary unions.
- **Group:** The category **Group** has groups as its objects, and group homomorphisms as its morphisms.
- **Graph:** The category **Graph** has graphs (undirected) as its objects, and graph morphisms (mapping vertices to vertices, preserving adjacency properties) as its morphisms. The category **DirGraph** has directed graphs as its objects, and the morphisms must now preserve adjacency as defined by a directed edge.
- **Poset:** The category **Poset** has partially ordered sets as its objects and order-preserving functions as its morphisms.
- **Meas:** The category **Meas** has measurable spaces as its objects and measurable functions as its morphisms. Recall that a measurable space $(\Omega, \mathcal{B})$ is defined by a set Ω and an associated σ-field of subsets B that is closed under complementation, and arbitrary unions and intersections, where the empty set $\emptyset \in \mathcal{B}$.

5.2 A Categorical Theory of Transformer Models

To illustrate the power of the simplicial sets and objects framework, we want to briefly explain how we can use it to define a novel hierarchical framework for generative AI, where each morphism $[m] \to [n]$ can be mapped into a Transformer module Vaswani et al. [2017]. It is straightforward to extend our discussion below to other building blocks of generative AI systems, including structured state space sequence models Gu et al. [2022] or image diffusion models Song and Ermon [2019]. As with all generative AI systems, the fundamental structure of a Transformer model is that it is a compositional structure made up of modular components, each of which computes a *permutation-equivariant* function over the vector space $\mathbb{R}^{d\times n}$ of n-length sequences of tokens, each embedded in a space of dimension d. We can define a commutative diagram showing the permutation equivariant property as shown below.

To begin with, following Fong et al. [2019], we can generically define a neural network layer of type (n_1, n_2) as a subset $C \subseteq [n_1] \times [n_2]$ where $n_1, n_2 \in \mathbb{N}$ are natural numbers, and $[n] = \{1, \ldots, n\}$. Notice how these can be viewed as the objects of a simplicial category Δ. These numbers n_1 and n_2 serve to define the number of inputs and outputs of each layer, C is a set of connections, and $(i, j) \in C$ means that node i is connected to node j in the network diagram. It is straightforward, but perhaps tedious, to define activation functions $\sigma : \mathbb{R} \to \mathbb{R}$ for each layer, but essentially each network layer defines a parameterized function $I : \mathbb{R}^{|C|+n_2} \times \mathbb{R}^{n_1} \to \mathbb{R}^{n_2}$, where the $\mathbb{R}|C|$ define the edge weights of each network edge and the $\mathbb{R}^{n_2}$ factor encodes individual unit biases. We can specialize these to Transformer models, in particular, noting that the Transfomer models compute specialized types of permutation-equivariant functions as defined by the commutative diagram below.



In the above commutative diagram, vertices are objects, and arrows are morphisms that define the action of a Transformer block. Here, $X \in \mathbb{R}^{d \times n}$ is a n-length sequence of tokens of dimensionality d. P is a permutation matrix. The function f computed by a Transformer block is such that $f(XP) = f(X)P$. This property is defined in the above diagram by setting $Y = f(X)P$, which can be computed in two ways, either first by permuting the input by the matrix P, and then applying f, or by

Let us understand the permutation equivariant property of the Transformer model in a bit more detail. Our notation for the Transformer model is based on Yun et al. [2020], although there are countless variations in the literature that we do not discuss further. These can be adapted into our categorical framework fairly straightforwardly based on the approach outlined below. Transformer models are inherently compositional, which makes them particularly convenient to model using category theory.

Definition 31. Yun et al. [2020], Vaswani et al. [2017] A **Transformer** block is a sequence-to-sequence function mapping $\mathbb{R}^{d \times n} \to \mathbb{R}^{d \times n}$. There are generally two layers: a *self-attention* layer Vaswani et al. [2017] and a token-wise feedforward layer. We assume tokens are embedded in a space of dimension d. Specifically, we model the inputs $X \in \mathbb{R}^{d \times n}$ to a Transformer block as n-length sequences of tokens in d dimensions, where each block computes the following function defined as $t^{h,m,r} : \mathbb{R}^{d \times n} : \mathbb{R}^{d \times n}$:

$$\begin{aligned} \text{Attn}(X) &= X + \sum_{i=1}^{h} W_O^i W_V^i X \cdot \sigma[W_K^i X)^T W_Q^i X] \\ \text{FF}(X) &= \text{Attn}(X) + W_2 \cdot \text{ReLU}(W_1 \cdot \text{Attn}(X) + b_1 \mathbf{1}_n^T, \end{aligned}$$

where $W_O^i \in \mathbb{R}^{d \times n}$, $W_K^i, W_Q^i, W_Q^i \in \mathbb{R}^{d \times n}$, $W_2 \in \mathbb{R}^{d \times r}$, $W_1 \in \mathbb{R}^{r \times d}$, and $b_1 \in \mathbb{R}^r$. The output of a Transformer block is $FF(X)$. Following convention, the number of "heads" is h, and each "head" size m are the principal parameters of the attention layer, and the size of the "hidden" feed-forward layer is r.

Transformer models take as input objects $X \in \mathbb{R}^{d \times n}$ representing n-length sequences of tokens in d dimensions, and act as morphisms that represent permutation equivariant functions $f : \mathbb{R}^{d \times n} \to \mathbb{R}^{d \times n}$ such that $f(XP) = f(X)P$ for any permutation matrix. Yun et al. [2020] show that the actual function computed by the Transformer model defined above is a permutation equivariant mapping.

Categories are compositional structures, which can be built out of smaller objects. Concretely, we define a category of transformers $\mathcal{C}_T$ where the objects are vectors $x \in \mathbb{R}^{d \times n}$ representing sequences of d-dimensional tokens of length n, and the composable arrows are *permutation-invariant* functions $\mathcal{T}^{h,m,r}$ comprised of a composition of transformer blocks $t^{h,m,r}$ of h heads of size m each, and a feedforward layer of r hidden nodes. Objects in a category interact with each other through arrows or morphisms. In the category $\mathcal{C}_T$ of Transformer models, the morphisms are the equivariant maps f by which one Transformer model block can be composed with another.

Definition 32. The category $\mathcal{C}_T$ of Transformer models is defined as follows:

- The objects Obj(C) are defined as vectors $X \in \mathbb{R}^{d \times n}$ denoting n-length sequences of tokens of dimension d.
- The arrows or morphisms of the category $\mathcal{C}_T$ are defined as a family of sequence-to-sequence functions and defined as:

$$T^{h,m,r} \coloneqq \{f : \mathbb{R}^{d \times n} \to \mathbb{R}^{d \times n} \mid \text{where } f(XP) = XP, \text{ for some permutation matrix } P\}$$

5.3 Constructing Simplicial Transformers from Transformer Categories

We now show how we can define a novel hierarchical theory of simplicial Transformers, first by embedding the category of Transformers into a simplicial set by computing the *nerve* of a functor that maps $\mathcal{C}_T$ into the simplicial set $S_\bullet^T$. The

nerve of a category $\mathcal{C}$ enables embedding $\mathcal{C}$ into the category of simplicial objects, which is a fully faithful embedding Lurie [2009], Richter [2020].

Definition 33. Let $\mathcal{F} : \mathcal{C} \to \mathcal{D}$ be a functor from category $\mathcal{C}$ to category $\mathcal{D}$. If for all arrows f the mapping $f \to Ff$

- injective, then the functor $\mathcal{F}$ is defined to be **faithful**.
- surjective, then the functor $\mathcal{F}$ is defined to be **full**.
- bijective, then the functor $\mathcal{F}$ is defined to be **fully faithful**.

Definition 34. The **nerve** of a category $\mathcal{C}$ is the set of composable morphisms of length n, for $n \geqslant 1$. Let $N_n(\mathcal{C})$ denote the set of sequences of composable morphisms of length n.

$$\{C_o \xrightarrow{f_1} C_1 \xrightarrow{f_2} \ldots \xrightarrow{f_n} C_n \mid C_i \text{ is an object in } \mathcal{C}, f_i \text{ is a morphism in } \mathcal{C}\}$$

The set of n-tuples of composable arrows in C, denoted by $N_n(\mathcal{C})$, can be viewed as a functor from the simplicial object $[n]$ to $\mathcal{C}$. Note that any nondecreasing map $\alpha : [m] \to [n]$ determines a map of sets $N_m(\mathcal{C}) \to N_n(\mathcal{C})$. The nerve of a category C is the simplicial set $N_\bullet : \Delta \to N_n(\mathcal{C})$, which maps the ordinal number object $[n]$ to the set $N_n(\mathcal{C})$.

The importance of the nerve of a category comes from a key result Lurie [2022], Richter [2020], showing it defines a full and faithful embedding of a category:

Theorem 3. The **nerve functor** $N_\bullet$: **Cat** $\to$ **Set** is fully faithful. More specifically, there is a bijection θ defined as:

$$\theta : \mathbf{Cat}(\mathcal{C}, \mathcal{C}') \to \mathbf{Set}_\Delta(N_\bullet(\mathcal{C}), N_\bullet(\mathcal{C}')$$

Unfortunately, the left adjoint to the nerve functor is not a full and faithful encoding of a simplicial set back into a suitable category. Note that a functor G from a simplicial object X to a category $\mathcal{C}$ can be lossy. For example, we can define the objects of $\mathcal{C}$ to be the elements of X_0, and the morphisms of $\mathcal{C}$ as the elements $f \in X_1$, where $f : a \to b$, and $d_0 f = a$, and $d_1 f = b$, and $s_0 a, a \in X$ as defining the identity morphisms $\mathbf{1}_a$. Composition in this case can be defined as the free algebra defined over elements of X_1, subject to the constraints given by elements of X_2. For example, if $x \in X_2$, we can impose the requirement that $d_1 x = d_0 x \circ d_2 x$. Such a definition of the left adjoint would be quite lossy because it only preserves the structure of the simplicial object X up to the 2-simplices. The right adjoint from a category to its associated simplicial object, in contrast, constructs a full and faithful embedding of a category into a simplicial set. In particular, the nerve of a category is such a right adjoint.

6 Layer 3 of GAIA: Universal Properties and the Category of Elements

A central and unifying principle in GAIA is that every pair of categorical layers is synchronized by a functor, along with a universal arrow. In this section, we introduce some additional ideas from category theory, including the fundamental Yoneda Lemma MacLane [1971] that states that all objects in a (generative AI) category can be defined in terms of their interactions. To understand the significance of this powerful lemma, keep in mind that it applies to any (small) category. In particular, we defined in Section 5 the category of Transformer models as equivariant functions over Euclidean spaces. What the Yoneda Lemma implies here is that any Transformer model building block can be defined (upto isomorphism) purely in terms of the interactions it makes with other Transformer building blocks. This somewhat strange parameterization provides deep insight into how objects in categories behave. In a concrete sense, Transformer models are based on defining words by their context in sentences, and an enriched form of the Yoneda Lemma can be directly applied to model the statistical representations of words learned by Transformers Bradley et al. [2022].

The bottom layer of GAIA is a (Grothendieck) category of elements Riehl [2017] that essentially "grounds" out the universal coalgebras at layer 2 of GAIA in terms of the concrete data that was used to build the foundation models in GAIA. Intuitively, Layer 3 stores the "training data" in a general relational structure. We first define how to construct the category of elements in a relational database. In particular, at the simplicial top layer, generative AI operators such as face and degeneracy operators define "graph surgery" Pearl [2009] operations on generative AI models, or in terms of "copy", "delete" operators in "string diagram surgery" defined on symmetric monoidal categories Jacobs et al. [2019]. These "surgery" operations at the next level may translate down to operations on probability distributions, measurable spaces, topological spaces, or chain complexes. This process follows a standard construction used widely in mathematics, for example group representations associate with any group G, a left $\mathbf{k}$-module M representation that enables modeling abstract group operations by operations on the associated modular representation. These concrete representations must satisfy the universal arrow property for them to be faithful.

6.1 Natural Transformations and Universal Arrows

Since we now can have multiple functors between the category `Para` and the category `Learn`, for example traditional backpropagation or a stochastic approximation "zeroth-order" approximation, we can compare these two functors using natural transformations. Given any two functors $F : C \to D$ and $G : C \to D$ between the same pair of categories, we can define a mapping between F and G that is referred to as a natural transformation. These are defined through a collection of mappings, one for each object c of C, thereby defining a morphism in D for each object in C.

Definition 35. Given categories C and D, and functors $F, G : C \to D$, a **natural transformation** $\alpha : F \Rightarrow G$ is defined by the following data:

- an arrow $\alpha_c : Fc \to Gc$ in D for each object $c \in C$, which together define the components of the natural transformation.
- For each morphism $f : c \to c'$, the following commutative diagram holds true:

$$\begin{array}{ccc} Fc & \xrightarrow{\alpha_c} & Gc \\ {\scriptstyle Ff}\downarrow & & \downarrow{\scriptstyle Gf} \\ Fc' & \xrightarrow[\alpha_{c'}]{} & Gc' \end{array}$$

A **natural isomorphism** is a natural transformation $\alpha : F \Rightarrow G$ in which every component α_c is an isomorphism.

A fundamental universal construction in category theory, called the *universal arrow* lies at the heart of many useful results, principally the Yoneda lemma that shows how object identity itself emerges from the structure of morphisms that lead into (or out of) it.

Definition 36. Given a functor $S : D \to C$ between two categories, and an object c of category C, a **universal arrow** from c to S is a pair $\langle r, u \rangle$, where r is an object of D and $u : c \to Sr$ is an arrow of C, such that the following universal property holds true:

- For every pair $\langle d, f \rangle$ with d an object of D and $f : c \to Sd$ an arrow of C, there is a unique arrow $f' : r \to d$ of D with $Sf' \circ u = f$.

Definition 37. If D is a category and $H : D \to \mathbf{Set}$ is a set-valued functor, a **universal element** associated with the functor H is a pair $\langle r, e \rangle$ consisting of an object $r \in D$ and an element $e \in Hr$ such that for every pair $\langle d, x \rangle$ with $x \in Hd$, there is a unique arrow $f : r \to d$ of D such that $(Hf)e = x$.

Theorem 4. Given any functor $S : D \to C$, the universal arrow $\langle r, u : c \to Sr \rangle$ implies a bijection exists between the **Hom** sets

$$\mathbf{Hom}_D(r, d) \simeq \mathbf{Hom}_C(c, Sd)$$

A special case of this natural transformation that transforms the identity morphism $\mathbf{1}_r$ leads us to the Yoneda lemma.

6.2 Yoneda Lemma

The Yoneda Lemma states that the set of all morphisms into an object d in a category $\mathcal{C}$, denoted as $\mathbf{Hom}_{\mathcal{C}}(-, d)$ and called the *contravariant functor* (or presheaf), is sufficient to define d up to isomorphism. The category of all presheaves forms a *category of functors*, and is denoted $\hat{\mathcal{C}} = \mathbf{Set}^{\mathcal{C}^{op}}$.We will briefly describe two concrete applications of this lemma to two important areas in AI and ML in this section: reasoning about causality and reasoning about distances. The Yoneda lemma plays a crucial role in this paper because it defines the concept of a *universal representation* in category theory. We first show that associated with universal arrows is the corresponding induced isomorphisms between **Hom** sets of morphisms in categories. This universal property then leads to the Yoneda lemma.

$$\begin{array}{ccc} D(r,r) & \xrightarrow{\phi_r} & C(c, Sr) \\ \downarrow{\scriptstyle D(r,f')} & & \downarrow{\scriptstyle C(c,Sf')} \\ D(r,d) & \xrightarrow{\phi_d} & C(c, Sd) \end{array}$$

As the two paths shown here must be equal in a commutative diagram, we get the property that a bijection between the **Hom** sets holds precisely when $\langle r, u : c \to Sr \rangle$ is a universal arrow from c to S. Note that for the case when the categories C and D are small, meaning their **Hom** collection of arrows forms a set, the induced functor $\mathbf{Hom}_C(c, S-)$ to **Set** is isomorphic to the functor $\mathbf{Hom}_D(r, -)$. This type of isomorphism defines a universal representation, and is at the heart of the causal reproducing property (CRP) defined below.

Lemma 1. Yoneda lemma: For any functor $F : C \to \mathbf{Set}$, whose domain category C is "locally small" (meaning that the collection of morphisms between each pair of objects forms a set), any object c in C, there is a bijection

$$\mathrm{Hom}(C(c, -), F) \simeq Fc$$

that defines a natural transformation $\alpha : C(c, -) \Rightarrow F$ to the element $\alpha_c(1_c) \in Fc$. This correspondence is natural in both c and F.

There is of course a dual form of the Yoneda Lemma in terms of the contravariant functor $C(-, c)$ as well using the natural transformation $C(-, c) \Rightarrow F$. A very useful way to interpret the Yoneda Lemma is through the notion of universal representability through a covariant or contravariant functor.

Definition 38. A **universal representation** of an object $c \in C$ in a category C is defined as a contravariant functor F together with a functorial representation $C(-, c) \simeq F$ or by a covariant functor F together with a representation $C(c, -) \simeq F$. The collection of morphisms $C(-, c)$ into an object c is called the **presheaf**, and from the Yoneda Lemma, forms a universal representation of the object.

Later in this paper, we will see how the Yoneda Lemma gives us a novel perspective on how to construct universal representers in non-symmetric generalized metric spaces that are essential to defining "nonsymmetric attention" in large language models.

A key distinguishing feature of category theory is the use of diagrammatic reasoning. However, diagrams are also viewed more abstractly as functors mapping from some indexing category to the actual category. Diagrams are useful in understanding universal constructions, such as limits and colimits of diagrams. To make this somewhat abstract definition concrete, let us look at some simpler examples of universal properties, including co-products and quotients (which in set theory correspond to disjoint unions). Coproducts refer to the universal property of abstracting a group of elements into a larger one.

Before we formally the concept of limit and colimits, we consider some examples. These notions generalize the more familiar notions of Cartesian products and disjoint unions in the category of **Sets**, the notion of meets and joins in the category **Preord** of preorders, as well as the least upper bounds and greatest lower bounds in lattices, and many other concrete examples from mathematics.

Example 11. If we consider a small "discrete" category $\mathcal{D}$ whose only morphisms are identity arrows, then the colimit of a functor $\mathcal{F} : \mathcal{D} \to \mathcal{C}$ is the *categorical coproduct* of $\mathcal{F}(D)$ for D, an object of category D, is denoted as

$$\mathrm{Colimit}_{\mathcal{D}} F = \bigsqcup_{D} \mathcal{F}(D)$$

In the special case when the category C is the category **Sets**, then the colimit of this functor is simply the disjoint union of all the sets $F(D)$ that are mapped from objects $D \in \mathcal{D}$.

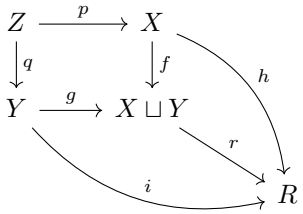
Example 12. Dual to the notion of colimit of a functor is the notion of *limit*. Once again, if we consider a small "discrete" category $\mathcal{D}$ whose only morphisms are identity arrows, then the limit of a functor $\mathcal{F} : \mathcal{D} \to \mathcal{C}$ is the *categorical product* of $\mathcal{F}(D)$ for D, an object of category D, is denoted as

$$\mathrm{limit}_{\mathcal{D}} F = \prod_{D} \mathcal{F}(D)$$

In the special case when the category C is the category **Sets**, then the limit of this functor is simply the Cartesian product of all the sets $F(D)$ that are mapped from objects $D \in \mathcal{D}$.

Category theory relies extensively on *universal constructions*, which satisfy a universal property. One of the central building blocks is the identification of universal properties through formal diagrams. Before introducing these definitions in their most abstract form, it greatly helps to see some simple examples.

We can illustrate the limits and colimits in diagrams using pullback and pushforward mappings.



An example of a universal construction is given by the above commutative diagram, where the coproduct object $X \sqcup Y$ uniquely factorizes any mapping $h : X \to R$, such that any mapping $i : Y \to R$, so that $h = r \circ f$, and furthermore $i = r \circ g$. Co-products are themselves special cases of the more general notion of co-limits. Figure 21 illustrates the fundamental property of a *pullback*, which along with *pushforward*, is one of the core ideas in category theory. The pullback square with the objects U, X, Y and Z implies that the composite mappings $g \circ f'$ must equal $g' \circ f$. In this example, the morphisms f and g represent a *pullback* pair, as they share a common co-domain Z. The pair of morphisms f', g' emanating from U define a *cone*, because the pullback square "commutes" appropriately. Thus, the pullback of the pair of morphisms f, g with the common co-domain Z is the pair of morphisms f', g' with common domain U. Furthermore, to satisfy the universal property, given another pair of morphisms x, y with common domain T, there must exist another morphism $k : T \to U$ that "factorizes" x, y appropriately, so that the composite morphisms $f'\ k = y$ and $g'\ k = x$. Here, T and U are referred to as *cones*, where U is the limit of the set of all cones "above" Z. If we reverse arrow directions appropriately, we get the corresponding notion of pushforward. So, in this example, the pair of morphisms f', g' that share a common domain represent a pushforward pair. As Figure 21, for any set-valued functor $\delta : S \to \mathbf{Sets}$, the Grothendieck category of elements $\int \delta$ can be shown to be a pullback in the diagram of categories. Here, $\mathbf{Set}_*$ is the category of pointed sets, and π is a projection that sends a pointed set $(X, x \in X)$ to its underlying set X.

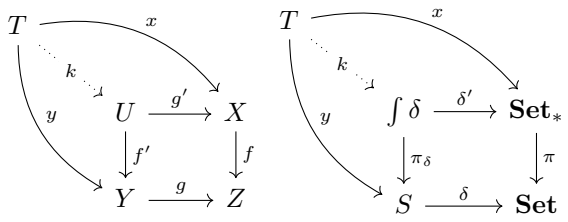
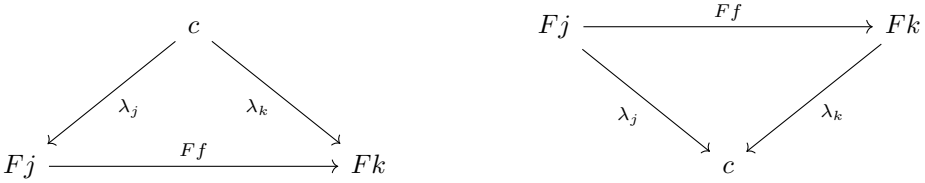


Figure 21: (**Left**) Universal Property of pullback mappings. (**Right**) The Grothendieck category of elements $\int \delta$ of any set-valued functor $\delta : S \to \mathbf{Set}$ can be described as a pullback in the diagram of categories. Here, $\mathbf{Set}_*$ is the category of pointed sets $(X, x \in X)$, and π is the "forgetful" functor that sends a pointed set $(X, x \in X)$ into the underlying set X.

We can now proceed to define limits and colimits more generally. We define a *diagram* F of *shape* J in a category C formally as a functor $F : J \to C$. We want to define the somewhat abstract concepts of *limits* and *colimits*, which will play a central role in this paper in identifying properties of AI and ML techniques. A convenient way to introduce these concepts is through the use of *universal cones* that are *over* and *under* a diagram.

For any object $c \in C$ and any category J, the *constant functor* $c : J \to C$ maps every object j of J to c and every morphism f in J to the identity morphisms 1_c. We can define a constant functor embedding as the collection of constant functors $\Delta : C \to C^J$ that send each object c in C to the constant functor at c and each morphism $f : c \to c'$ to the constant natural transformation, that is, the natural transformation whose every component is defined to be the morphism f.

Definition 39. A **cone over** a diagram $F : J \to C$ with the **summit** or **apex** $c \in C$ is a natural transformation $\lambda : c \Rightarrow F$ whose domain is the constant functor at c. The components $(\lambda_j : c \to Fj)_{j \in J}$ of the natural transformation can be viewed as its **legs**. Dually, a **cone under** F with **nadir** c is a natural transformation $\lambda : F \Rightarrow c$ whose legs are the components $(\lambda_j : Fj \to c)_{j \in J}$.



Cones under a diagram are referred to usually as *cocones*. Using the concept of cones and cocones, we can now formally define the concept of limits and colimits more precisely.

Definition 40. For any diagram $F : J \to C$, there is a functor

$$\text{Cone}(-, F) : C^{op} \to \textbf{Set}$$

which sends $c \in C$ to the set of cones over F with apex c. Using the Yoneda Lemma, a **limit** of F is defined as an object $\lim F \in C$ together with a natural transformation $\lambda : \lim F \to F$, which can be called the **universal cone** defining the natural isomorphism

$$C(-, \lim F) \simeq \text{Cone}(-, F)$$

Dually, for colimits, we can define a functor

$$\text{Cone}(F, -) : C \to \textbf{Set}$$

that maps object $c \in C$ to the set of cones under F with nadir c. A **colimit** of F is a representation for $\text{Cone}(F, -)$. Once again, using the Yoneda Lemma, a colimit is defined by an object $\text{Colim}F \in C$ together with a natural transformation $\lambda : F \to \text{colim}F$, which defines the **colimit cone** as the natural isomorphism

$$C(\text{colim}F, -) \simeq \text{Cone}(F, -)$$

Limit and colimits of diagrams over arbitrary categories can often be reduced to the case of their corresponding diagram properties over sets. One important stepping stone is to understand how functors interact with limits and colimits.

Definition 41. For any class of diagrams $K : J \to C$, a functor $F : C \to D$

- **preserves** limits if for any diagram $K : J \to C$ and limit cone over K, the image of the cone defines a limit cone over the composite diagram $FK : J \to D$.
- **reflects** limits if for any cone over a diagram $K : J \to C$ whose image upon applying F is a limit cone for the diagram $FK : J \to D$ is a limit cone over K
- **creates** limits if whenever $FK : J \to D$ has a limit in D, there is some limit cone over FK that can be lifted to a limit cone over K and moreoever F reflects the limits in the class of diagrams.

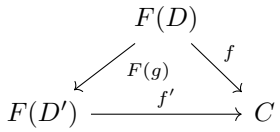
To interpret these abstract definitions, it helps to concretize them in terms of a specific universal construction, like the pullback defined above $c' \to c \leftarrow c''$ in C. Specifically, for pullbacks:

- A functor F **preserves pullbacks** if whenever p is the pullback of $c' \to c \leftarrow c''$ in C, it follows that Fp is the pullback of $Fc' \to Fc \leftarrow Fc''$ in D.
- A functor F **reflects pullbacks** if p is the pullback of $c' \to c \leftarrow c''$ in C whenever Fp is the pullback of $Fc' \to Fc \leftarrow Fc''$ in D.
- A functor F **creates pullbacks** if there exists some p that is the pullback of $c' \to c \leftarrow c''$ in C whenever there exists a d such that d is the pullback of $Fc' \to Fc \leftarrow Fc''$ in F.

Universality of Diagrams

In the category **Sets**, we know that every object (i.e., a set) X can be expressed as a coproduct (i.e., disjoint union) of its elements $X \simeq \sqcup_{x \in X}\{x\}$, where $x \in X$. Note that we can view each element $x \in X$ as a morphism $x : \{*\} \to X$ from the one-point set to X. The categorical generalization of this result is called the *density theorem* in the theory of sheaves. First, we define the key concept of a *comma category*.

Definition 42. Let $F : \mathcal{D} \to \mathcal{C}$ be a functor from category $\mathcal{D}$ to $\mathcal{C}$. The **comma category** $F \downarrow \mathcal{C}$ is one whose objects are pairs (D, f), where $D \in \mathcal{D}$ is an object of $\mathcal{D}$ and $f \in \textbf{Hom}_{\mathcal{C}}(F(D), C)$, where C is an object of $\mathcal{C}$. Morphisms in the comma category $F \downarrow \mathcal{C}$ from (D, f) to (D', f'), where $g : D \to D'$, such that $f' \circ F(g) = f$. We can depict this structure through the following commutative diagram:



We first introduce the concept of a *dense* functor:

Definition 43. Let D be a small category, C be an arbitrary category, and $F : \mathcal{D} \to \mathcal{D}$ be a functor. The functor F is **dense** if for all objects C of $\mathcal{C}$, the natural transformation

$$\psi_F^C : F \circ U \to \Delta_C, \;\; (\psi_F^C)_{(\mathcal{D},f)} = f$$

is universal in the sense that it induces an isomorphism $\text{Colimit}_{F\downarrow C} F \circ U \simeq C$. Here, $U : F \downarrow C \to \mathcal{D}$ is the projection functor from the comma category $F \downarrow \mathcal{C}$, defined by $U(D, f) = D$.

A fundamental consequence of the category of elements is that every object in the functor category of presheaves, namely contravariant functors from a category into the category of sets, is the colimit of a diagram of representable objects, via the Yoneda lemma. Notice this is a generalized form of the density notion from the category **Sets**.

Theorem 5. Universality of Diagrams: In the functor category of presheaves $\mathbf{Set}^{\mathcal{C}^{op}}$, every object P is the colimit of a diagram of representable objects, in a canonical way.

6.3 Universal Arrows and Elements

We explore the universal arrow property more deeply in this section, showing how it provides the conceptual basis behind the (metric) Yoneda Lemma, and Grothendieck's category of elements.

A special case of the universal arrow property is that of universal element, which as we will see below plays an important role in the GAIA architecture in defining a suitably augmented category of elements, based on a construction introduced by Grothendieck.

Definition 44. If D is a category and $H : D \to \mathbf{Set}$ is a set-valued functor, a **universal element** associated with the functor H is a pair $\langle r, e\rangle$ consisting of an object $r \in D$ and an element $e \in Hr$ such that for every pair $\langle d, x\rangle$ with $x \in Hd$, there is a unique arrow $f : r \to d$ of D such that $(Hf)e = x$.

Example 13. Let E be an equivalence relation on a set S, and consider the quotient set S/E of equivalence classes, where $p : S \to S/E$ sends each element $s \in S$ into its corresponding equivalence class. The set of equivalence classes S/E has the property that any function $f : S \to X$ that respects the equivalence relation can be written as $fs = fs'$ whenever $s \sim_E s'$, that is, $f = f' \circ p$, where the unique function $f' : S/E \to X$. Thus, $\langle S/E, p\rangle$ is a universal element for the functor H.

6.4 The Category of Elements

We turn next to define the category of elements, based on a construction by Grothendieck, and illustrate how it can serve as the basis for inference at each layer of the UCLA architecture. In particular, Spivak [2013] shows how the category of elements can be used to define SQL queries in a relational database.

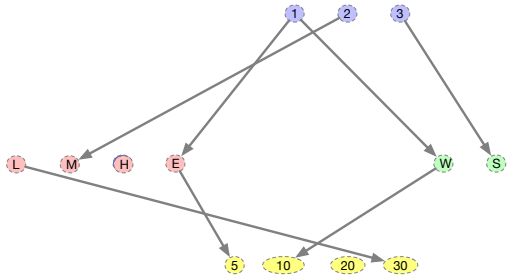
Definition 45. Given a set-valued functor $\delta : \mathcal{C} \to \mathbf{Set}$ from some category $\mathcal{C}$, the induced **category of elements** associated with δ is a pair $(\int \delta, \pi_\delta)$, where $\int \delta \in$ **Cat** is a category in the category of all categories **Cat**, and $\pi_\delta : \int \delta \to \mathcal{C}$ is a functor that "projects" the category of elements into the corresponding original category $\mathcal{C}$. The objects and arrows of $\int \delta$ are defined as follows:

- $\text{Ob}(\int \delta) = \{(s, x) | x \in \text{Ob}(\rfloor), x \in \delta s\}$.
- $\mathbf{Hom}_{\int \delta}((s, x), (s', x')) = \{f : s \to s' | \delta f(x) = x'\}$

Example 14. To illustrate the category of elements construction, let us consider the toy climate change DAG model shown in Figure 22. Let the category C be defined by this DAG model, where the objects Ob(C) are defined by the four vertices, and the arrows $\mathbf{Hom}_{\mathcal{C}}$ are defined by the four edges in the model. The set-valued functor $\delta : \mathcal{C} \to \mathbf{Set}$ maps each object (vertex) in C to a set of instances, thereby turning the causal DAG model into an associated set of tables. For example, **Climate Change** is defined as a table of values, which could be modeled as a multinomial variable taking on a set of discrete values, and for each of its values, the arrow from **Climate Change** to **Rainfall** maps each specific value of **Climate Change** to a value of **Rainfall**, thereby indicating a causal effect of climate change on the amount of rainfall in California. Im the figure, **Climate Change** is mapped to three discrete levels (marked 1, 2 and 3). Rainfall

amounts are discretized as well into low (marked "L"), medium (marked "M"), high (marked "H"), or extreme (marked "E"). Wind speeds are binned into two levels (marked "W" for weak, and "S" for strong). Finally, the percentage of California wildfires is binned between 5 to 30. Not all arrows that exist in the Grothendieck category of elements are shown, for clarity.

Figure 22: A toy DAG model of climate change to illustrate the category of elements construction.



Many properties of Grothendieck's construction can be exploited (some of these are discussed in the context of relational database queries in Spivak [2013]), but for our application, we are primarily interested in the associated class of lifting problems that define queries in a generative AI model.

6.5 Lifting Problems in Generative AI

Definition 46. If S is a collection of morphisms in category $\mathcal{C}$, a morphism $f : A \to B$ has the **left lifting property with respect to S** if it has the left lifting property with respect to every morphism in S. Analogously, we say a morphism $p : X \to Y$ has the **right lifting property with respect to S** if it has the right lifting property with respect to every morphism in S.

Many properties of Grothendieck's construction can be exploited (some of these are discussed in the context of relational database queries in Spivak [2013]), but for our application to generative AI, we are primarily interested in the associated class of lifting problems that can be used to define queries and build foundation models.

Definition 47. If S is a collection of morphisms in category $\mathcal{C}$, a morphism $f : A \to B$ has the **left lifting property with respect to S** if it has the left lifting property with respect to every morphism in S. Analogously, we say a morphism $p : X \to Y$ has the **right lifting property with respect to S** if it has the right lifting property with respect to every morphism in S.

We now turn to sketch some examples of the application of lifting problems for generative AI. Many problems in causal inference on graphs involve some particular graph property. To formulate it as a lifting problem, we will use the following generic template, following the initial application of lifting problems to database queries proposed by Spivak [2013].

$$\begin{array}{ccc} Q & \xrightarrow{\mu} & \int \delta \\ \downarrow{\scriptstyle f} & \overset{h}{\nearrow} & \downarrow{\scriptstyle p} \\ R & \xrightarrow{\nu} & \mathcal{C} \end{array}$$

Here, Q is a generic query that we want answered, which could range from a database query, as in the original setting studied by Spivak [2013], but more interestingly, it could be a particular graph property relating to generative AI. By suitably modifying the base category, the lifting problem formulation can be used to encode a diverse variety of problems in generative AI inference. R represents a fragment of the complete generative AI model $\mathcal{C}$, and δ is the category of elements defined above. Finally, h gives all solutions to the lifting problem.

Example 15. Consider the category of directed graphs defined by the category $\mathcal{G}$, where Ob($\mathcal{G}$) = {V, E}, and the morphisms of $\mathcal{G}$ are given as $\mathbf{Hom}_{\mathcal{G}}$ = {s, t}, where $s : E \to V$ and $t : E \to V$ define the source and terminal nodes of each vertex. Then, the category of all directed graphs is precisely defined by the category of all functors $\delta : \mathcal{G} \to \mathbf{Set}$. Any particular graph is defined by the functor $X : \mathcal{G} \to \mathbf{Set}$, where the function $X(s) : X(E) \to X(V)$ assigns to every edge its source vertex. For causal inference, we may want to check some property of a graph, such as the property that every vertex in X is the source of some edge. The following lifting problem ensures that every vertex has a source

edge in the graph. The category of elements $\int \delta$ shown below refers to a construction introduced by Grothendieck, which will be defined in more detail later.

$$\begin{array}{ccc} V(\bullet) & \xrightarrow{\mu} & \int \delta \\ \downarrow f & \overset{h}{\nearrow} & \downarrow p \\ \{E(\bullet) \overset{s}{\to} V(\bullet)\} & \xrightarrow{\nu} & \mathcal{G} \end{array}$$

Example 16. As another example of the application of lifting problems to causal inference, let us consider the problem of determining whether two causal DAGs, G_1 and G_2 are Markov equivalent Andersson et al. [1997]. A key requirement here is that the immoralities of G_1 and G_2 must be the same, that is, if G_1 has a collider $A \to B \leftarrow C$, where there is no edge between A and C, then G_2 must also have the same collider, and none others. We can formulate the problem of finding colliders as the following lifting problem. Note that the three vertices A, B and C are bound to an actual graph instance through the category of elements $\int \delta$ (as was illustrated above), using the top right morphism μ. The bottom left morphism f binds these three vertices to some collider. The bottom right morphism ν requires this collider to exist in the causal graph $\mathcal{G}$ with the same bindings as found by μ. The dashed morphisms h finds all solutions to this lifting problem, that is, all colliders involving the vertices A, B and C.

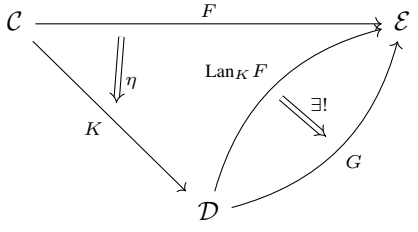
$$\begin{array}{ccc} \{A(\bullet), B(\bullet), C(\bullet)\} & \xrightarrow{\mu} & \int \delta \\ \downarrow f & \overset{h}{\nearrow} & \downarrow p \\ \{A(\bullet) \to B(\bullet) \leftarrow C(\bullet)\} & \xrightarrow{\nu} & \mathcal{G} \end{array}$$

If the category of elements is defined by a functor mapping a database schema into a table of instances, then the associated lifting problem corresponds to familiar problems like SQL queries in relational databases Spivak [2013]. In our application, we can use the same machinery to formulate causal inference queries by choosing the categories appropriately. To complete the discussion, we now make the connection between universal arrows and the core notion of universal representations via the Yoneda Lemma.

6.6 Kan Extension

It is well known in category theory that ultimately every concept, from products and co-products, limits and co-limits, and ultimately even the Yoneda Lemma (see below), can be derived as special cases of the Kan extension [MacLane, 1971]. Kan extensions intuitively are a way to approximate a functor $\mathcal{F}$ so that its domain can be extended from a category $\mathcal{C}$ to another category $\mathcal{D}$. Because it may be impossible to make commutativity work in general, Kan extensions rely on natural transformations to make the extension be the best possible approximation to $\mathcal{F}$ along $\mathcal{K}$. We want to briefly show Kan extensions can be combined with the category of elements defined above to construct "migration functors" that map from one generative AI model into another. These migration functors were originally defined in the context of database migration Spivak [2013], but can also be applied to generative AI inference. By suitably modifying the category of elements from a set-valued functor $\delta : \mathcal{C} \to \mathbf{Set}$, to some other category, such as the category of topological spaces, namely $\delta : \mathcal{C} \to \mathbf{Top}$, we can extend the migration functors into solving more abstract generative AI inference questions. Here, for simplicity, we restrict our focus to Kan extensions for migration functors over the category of elements defined over instances of a generative AI model.

Definition 48. A **left Kan extension** of a functor $F : \mathcal{C} \to \mathcal{E}$ along another functor $K : \mathcal{C} \to \mathcal{D}$, is a functor $\text{Lan}_K F : \mathcal{D} \to \mathcal{E}$ with a natural transformation $\eta : F \to \text{Lan}_F \circ K$ such that for any other such pair $(G : \mathcal{D} \to \mathcal{E}, \gamma : F \to GK)$, γ factors uniquely through η. In other words, there is a unique natural transformation $\alpha : \text{Lan}_F \implies G$.



A **right Kan extension** can be defined similarly. To understand the significance of Kan extensions for causal inference, we note that under a causal intervention, when a causal category S gets modified to T, evaluating the modified generative AI model over a database of instances can be viewed as an example of Kan extension.

Let $\delta : S \to \mathbf{Set}$ denote the original generative AI model defined by the category S with respect to some dataset. Let $\epsilon : T \to \mathbf{Set}$ denote the effect of some change in the category S to T, such as deletion of a morphism, as illustrated in Figure 23. Intuitively, we can consider three cases: the *pullback* Δ_F along F, which maps the effect of a deletion back to the original model, the *left pushforward* Σ_F and the *right pushforward* $\prod_F$, which can be seen as adjoints to the pullback Δ_F.

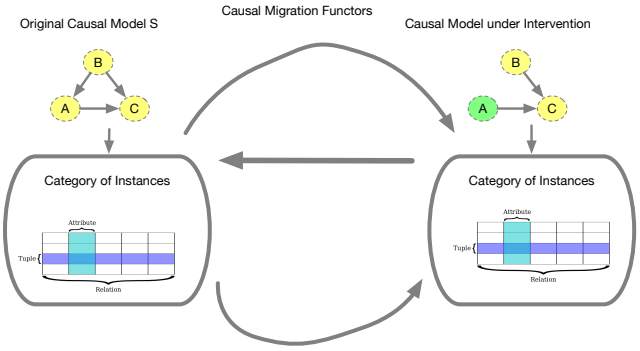


Figure 23: Kan extensions are useful in modeling the effects of modifications of generative AI models, such as deletion of morphisms, where in this toy example of a model over three objects A, B, and C, the object A is intervened upon, eliminating the morphism into it from object B.

Following Spivak [2013], we can define three *migration functors* that evaluate the impact of a modification of a generative AI model with respect to a dataset of instances.

1. The functor $\Delta_F : \epsilon \to \delta$ sends the functor $\epsilon : T \to \mathbf{Set}$ to the composed functor $\delta \circ F : S \to \mathbf{Set}$.
2. The functor $\Sigma_F : \delta \to \epsilon$ is the left Kan extension along F, and can be seen as the left adjoint to Δ_F.
   The functor $\prod_F : \delta \to \epsilon$ is the right Kan extension along F, and can be seen as the right adjoint to Δ_F.

To understand how to implement these functors, we use the following proposition that is stated in Spivak [2013] in the context of database queries, which we are restating in the setting of generative AI.

Theorem 6. Let $F : S \to T$ be a functor. Let $\delta : S \to \mathbf{Set}$ and $\epsilon : T \to \mathbf{Set}$ be two set-valued functors, which can be viewed as two instances of a generative AI model defined by the category S and T. If we view T as the generative AI category that results from a modification caused by some modification on S (e.g., deletion of an edge), then there is a commutative diagram linking the category of elements between S and T.

$$\begin{array}{ccc} \int \delta & \longrightarrow & \int \epsilon \\ \downarrow{\scriptstyle \pi_\delta} & & \downarrow{\scriptstyle \pi_\epsilon} \\ S & \xrightarrow{F} & T \end{array}$$

Proof. To check that the above diagram is a pullback, that is, $\int \delta \simeq S \times_T \int \delta$, or in words, the fiber product, we can check the existence of the pullback component wise by comparing the set of objects and the set of morphisms in $\int \delta$ with the respective sets in $S \times_T \int \epsilon$. □

For simplicity, we defined the migration functors above with respect to an actual dataset of instances. More generally, we can compose the set-valued functor $\delta : S \to \mathbf{Set}$ with a functor $\mathcal{T} : \mathbf{Set} \to \mathbf{Top}$ to the category of topological spaces to derive a Kan extension formulation of the definition of an intervention. We discuss this issue in Section 8 on homotopy in generative AI.

6.7 The Metric Yoneda Lemma

One disadvantage of current generative AI systems, such as large language models, is that they are based a symmetric model of distances. The Yoneda Lemma MacLane [1971], one of the most celebrated results in category theory, can be

used to build universal representers in non-symmetric generalized metric spaces, leading to a metric Yoneda Lemma Bonsangue et al. [1998]. Stated in simple terms, the Yoneda Lemma states the mathematical objects are determined (up to isomorphism) by the interactions they make with other objects in a category. We will show the surprising results of applying this lemma to problems involving computing distances between objects in a metric space.

A general principle in machine learning (see Figure 24) to discriminate two objects (e.g., probability distributions, images, text documents etc.) is to compare them in a suitable metric space. We now describe a category of generalized metric spaces, where a metric form of the Yoneda Lemma gives us surprising insight. Often, in category theory, we want to work in an enriched category. One of the most interesting ways to design categories for applications in AI and ML is to look to augment the basic structure of a category with additional properties. For example, the collection of morphisms from an object x to an object y in a category $\mathcal{C}$ often has additional structure, besides just being a set. Often, it satisfies additional properties, such as forming a space of some kind such as a vector space or a topological space. We can think of such categories as *enriched* categories that exploit some desirable properties. We will illustrate one such example of primary importance to applications in AI and ML that involve measuring the distance between two objects. A distance function is assumed to return some non-negative value between 0 and ∞, and we will view distances as defining enriched $[0, \infty]$ categories. We summarize some results here from Bonsangue et al. [1998].

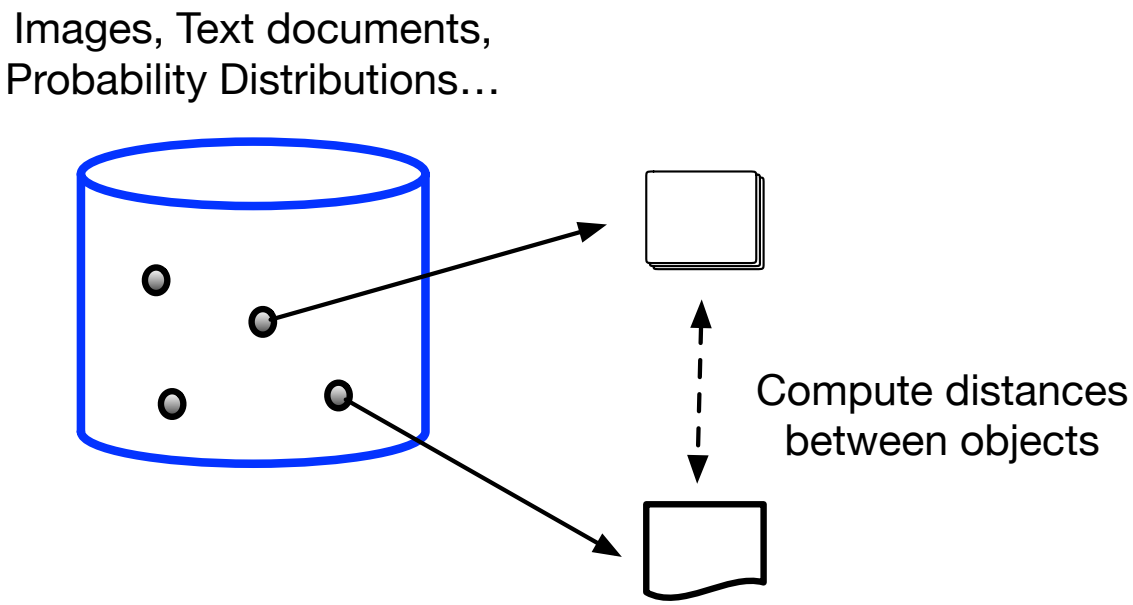


Figure 24: Many algorithms in AI and ML involve computing distances between objects in a *metric space*. Interpreting distances categorically leads to powerful ways to reason about generalized metric spaces.

Figure 24 illustrates a common motif among many AI and ML algorithms: define a problem in terms of computing distances between a group of objects. Examples of objects include points in n-dimensional Euclidean space, probability distributions, text documents represented as strings of tokens, and images represented as matrices. More abstractly, a *generalized metric space* (X, d) is a set X of objects, and a non-negative function $X(-,-) : X \times X \to [0, \infty]$ that satisfies the following properties:

1. $X(x, x) = 0$: distance between the same object and itself is 0.
2. $X(x, z) \leqslant X(x, y) + X(y, z)$: the famous *triangle inequality* posits that the distance between two objects cannot exceed the sum of distances between each of them and some other intermediate third object.

In particular, generalized metric spaces are not required to be *symmetric*, or satisfy the property that if the distance between two objects x and y is 0 implies x must be identical to y, or finally that distances must be finite. These additional three properties listed below are what defines the usual notion of a *metric* space:

1. If $X(x, y) = 0$ and $X(y, x) = 0$ then $x = y$.
2. $X(x, y) = X(y, x)$.
3. $X(x, y) < \infty$.

In fact, we can subsume the previous discussion of causal inference under the notion of generalized metric spaces by defining a category around *preorders* $(P, \leqslant)$, which are relations that are reflexive and transitive, but not symmetric. Causal inference fundamentally involves constructing a preorder over the set of variables in a domain. Here are some examples of generalized metric spaces:

1. Any preorder $(P, \leqslant)$ such that all $p, q, r \in P$, if $p \leqslant q$ and $q \leqslant r$, then, $p \leqslant r$, and $p \leqslant p$, where

$$P(p,q) = \left\{ \begin{array}{ccc} 0 & \text{if} & p \leqslant q \\ \infty & \text{if} & p \not\leqslant q \end{array} \right\}$$

2. The set of strings Σ^* over some alphabet defined as the set Σ where the distance between two strings u and v is defined as

$$\Sigma^*(u,v) = \left\{ \begin{array}{ccc} 0 & \text{if} & u \text{ is a prefix of } v \\ 2^{-n} & \text{otherwise} & \text{where } n \text{ is the longest common prefix of } u \text{ and } v \end{array} \right\}$$

3. The set of non-negative distances $[0, \infty]$ where the distance between two objects u and v is defined as

$$[0,\infty](u,v) = \left\{ \begin{array}{ccc} 0 & \text{if} & u \geqslant v \\ v - u & \text{otherwise} & \text{where } r < s \end{array} \right\}$$

4. The powerset $\mathcal{P}(X)$ of all subsets of a standard metric space, where the distance between two subsets $V, W \subseteq X$ is defined as

$$\mathcal{P}(X)(V,W) = \inf\{\epsilon > 0 | \forall v \in V, \exists w \in W, X(v,w) \leqslant \epsilon\}$$

which is often referred to as the *non-symmetric Hausdorff distance*.

Generalized metric spaces can be shown to be $[0, \infty]$-enriched categories as the collection of all morphisms between any two objects itself defines a category. In particular, the category $[0, \infty]$ is a complete and co-complete symmetric monoidal category. It is a category because objects are the non-negative real numbers, including ∞, and for two objects r and s in $[0, \infty]$, there is an arrow from r to s if and only if $r \leqslant s$. It is complete and co-complete because all equalizers and co-equalizers exist as there is at most one arrow between any two objects. The categorical product $r \sqcap s$ of two objects r and s is simply $\max\{r, s\}$, and the categorical coproduct $r \sqcup s$ is simply $\min\{r, s\}$. More generally, products are defined by supremums, and coproducts are defined by infimums. Finally, the *monoidal* structure is induced by defining the tensoring of two objects through "addition":

$$+ : [0,\infty] \times [0,\infty] \to [0,\infty]$$

where $r + s$ is simply their sum, and where as usual $r + \infty = \infty + r = \infty$.

The category $[0, \infty]$ is also a *compact closed* category, which turns out to be a fundamentally important property, and can be simply explained in this case as follows. We can define an "internal hom functor" $[0, \infty](-, -)$ between any two objects r and s in $[0, \infty]$ the distance $[0, \infty]$ as defined above, and the *co pre-sheaf* $[0, \infty](t, -)$ is *right adjoint* to $t + -$ for any $t \in [0, \infty]$.

Theorem 7. For all r, s and $t \in [0, \infty]$,

$$t + s \geqslant r \quad \text{if and only if} \quad s \geqslant [0,\infty](t,r)$$

We will explain the significance of compact closed categories for reasoning about AI and ML systems in more detail later, but in particular, we note that reasoning about feedback requires using compact closed categories to represent "dual" objects that are diagrammatically represented by arrows that run in the "reverse" direction from right to left (in addition to the usual convention of information flowing from left to right from inputs to outputs in any process model).

We can also define a category of generalized metric spaces, where each generalized metric space itself as an object, and for the morphism between generalized metric spaces X and Y, we can choose a *non-expansive function* $f : X \to Y$ which has the *contraction property*, namely

$$Y(f(x), f(y)) \leqslant c \cdot X(x,y)$$

where $0 < c < 1$ is assumed to be some real number that lies in the unit interval. The category of generalized metric spaces will turn out to be of crucial importance in this paper as we will use a central result in category theory – the Yoneda Lemma – to give a new interpretation to distances.

Finally, let us state a "metric" version of the Yoneda Lemma specifically for the case of $[0, \infty]$-enriched categories in generalized metric spaces:

Theorem 8. Bonsangue et al. [1998] (**Yoneda Lemma for generalized metric spaces**): Let X be a generalized metric space. For any $x \in X$, let

$$X(-, x) : X^{\text{op}} \to [0, \infty], \;\; y \longmapsto X(y, x)$$

Intuitively, what the generalized metric version of the Yoneda Lemma is stating is that it is possible to represent an element of a generalized metric space by its co-presheaf, exactly analogous to what we will see below in the next section for causal inference! If we use the notation

$$\hat{X} = [0, \infty]^{X^{\text{op}}}$$

to indicate the set of all non-expansive functions from X^{op} to $[0, \infty]$, then the Yoneda embedding defined by $y \longmapsto X(y, x)$ is in fact a non-expansive function, and itself an element of $\hat{X}$! Thus, it follows from the general Yoneda Lemma that for any other element ϕ in $\hat{X}$,

$$\hat{X}(X(-, x), \phi) = \phi(x)$$

Another fundamental result is that the Yoneda embedding for generalized metric spaces is an *isometry*. Again, this is exactly analogous to what we see below for causal inference, which we will denote as the causal reproducing property.

Theorem 9. The Yoneda embedding $y : X \to \hat{X}$, defined for $x \in X$ by $y(x) = X(-, x)$ is *isometric*, that is, for all $x, x' \in X$, we have:

$$X(x, x') = \hat{X}(y(x), y(x')) = \hat{X}(X(-, x), X(-, x'))$$

Once again, we will see a remarkable resemblance of this result to the Causal Representer Theorem below. With the metric Yoneda Lemma in hand, we can now define a framework for solving static UIGs in generalized metric spaces.

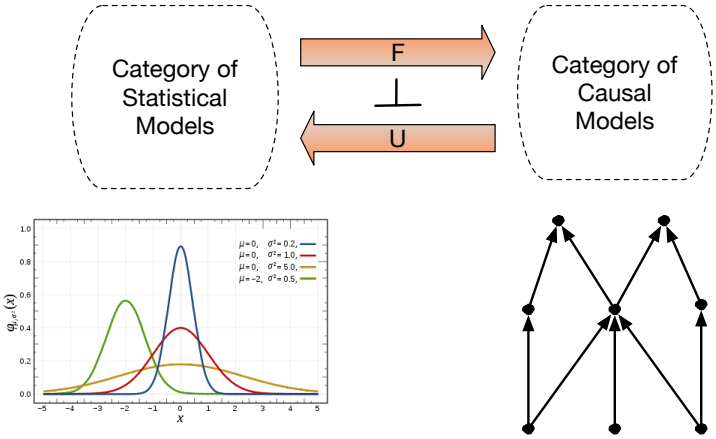
Definition 49. Two objects c and d are isomorphic in a generalized metric space category X if they are isometrically mapped into the category $\hat{X}$ by the Yoneda embedding $c \to X(-, c)$ and $d \to X(-, d)$ such that $X(c, d) = \hat{X}(X(-, c), X(-, d))$, where they can be defined isomorphically by a suitable pair of suitable natural transformations.

6.8 Adjoint Functors

Adjoint functors naturally arise in a number of contexts, among the most important being between "free" and "forgetful" functors. Let us consider a canonical example that is of prime significance in many applications in AI and ML.

Figure 25 provides a high level overview of the relationship between a category of statistical generative AI models and a category of causal generative AI models that can be seen as being related by a pair of adjoint "forgetful-free" functors. A statistical model can be abstractly viewed in terms of its conditional independence properties. More concretely, the category of *separoids*, defined in Section 2, consists of objects called separoids $(S, \leqslant)$, which are semilattices with a preordering $\leqslant$ where the elements $x, y, z \in S$ denote entities in a statistical model. We define a ternary relation $(\bullet \perp \bullet | \bullet) \subseteq S \times S \times S$, where $(x \perp y | z)$ is interpreted as the statement x is conditionally independent of y given z to denote a relationship between triples that captures abstractly the property that occurs in many applications in AI and ML. For example, in statistical ML, a sufficient statistic $T(X)$ of some dataset X, treated as a random variable, is defined to be any function for which the conditional independence relationship $(X \perp \theta | T(X))$, where $\theta \in \mathbb{R}^k$ denotes the parameter vector of some statistical model $P(X)$ that defines the true distribution of the data. Similarly, in causal inference, $(x \perp y | z) \Rightarrow p(x, y, z) = p(x|z)p(y|z)$ denotes a statement about the probabilistic conditional independence of x and y given z. In causal inference, the goal is to recover a partial order defined as a directed acyclic graph (DAG) that ascribes causality among a set of random variables from a dataset specifying a sample of their joint distribution. It is well known that without non-random interventions, causality cannot be inferred uniquely, since because of Bayes rule, there is no way to distinguish causal generative AI models such as $x \to y \to z$ from the reverse relationship $z \to y \to x$. In both these models, $x \perp z | y$ and because of Bayes inversion, one model can be recovered from the other.

Figure 25: Adjoint functors provide an elegant characterization of the relationship between the category of statistical generative AI models and that of causal generative AI models. Statistical models can be viewed as the result of applying a “forgetful" functor to a causal model that drops the directional structure in a causal model, whereas causal models can be viewed as “words" in a “free" algebra that results from the left adjoint functor to the forgetful functor.



We can define a “free-forgetful" pair of adjoint functors between the category of conditional independence relationships, as defined by separoid objects, and the category of causal generative AI models parameterized by DAG models.

We first review some basic material relating to adjunctions defined by adjoint functors, before proceeding to describe the theory of monads, as the two are intimately related. Our presentation of adjunctions and monads is based on Riehl’s excellent textbook on category theory Riehl [2017] to which the reader is referred to for a more detailed explanation. Adjunctions are defined by an opposing pair of functors $F : C \leftrightarrow D : G$ that can be defined more precisely as follows.

Definition 50. An **adjunction** consists of a pair of functors $F : C \to D$ and $G : D \to C$, where F is often referred to *left adjoint* and G is referred to as the *right adjoint*, that result in the following isomorphism relationship holding between their following sets of homomorphisms in categories C and D:

$$D(Fc, d) \simeq C(c, Gd)$$

We can express the isomorphism condition more explicitly in the form of the following commutative diagram:

$$\begin{array}{ccc} D(Fc,d) & \xrightarrow{\simeq} & C(c,Gd) \\ \downarrow{\scriptstyle k_*} & & \downarrow{\scriptstyle Gk_*} \\ D(Fc,d') & \xrightarrow{\simeq} & C(c,Gd') \end{array}$$

Here, $k : d \to d'$ is any morphism in D, and k_* denotes the “pullback" of k with the mapping $f : Fc \to d$ to yield the composite mapping $k \circ f$. The adjunction condition holds that the transpose of this composite mapping is equal to the composite mapping $g : c \to Gd$ with $Gk : Gd \to Gd'$. We can express this dually as well, as follows:

$$\begin{array}{ccc} D(Fc,d) & \xrightarrow{\simeq} & C(c,Gd) \\ \downarrow{\scriptstyle Fh^*} & & \downarrow{\scriptstyle h^*} \\ D(Fc',d) & \xrightarrow{\simeq} & C(c',Gd') \end{array}$$

where now $h : c' \to c$ is a morphism in C, and h^* denote the “pushforward" of h. Once again, the adjunction condition is a statement that the transpose of the composite mapping $f \circ Fh : Fc' \to d$ is identical to the composite of the mappings $h : c \to c'$ with $f : c \to Gd$.

It is common to denote adjoint functors in this turnstile notation, indicating that $F : C \to D$ is left adjoint to $G : D \to C$, or more simply as $F \vdash G$.

$$\mathcal{D} \underset{F}{\overset{G}{\underset{\longleftarrow}{\overset{\longrightarrow}{\top}}}} \mathcal{C}.$$

We can use the concept of universal arrows introduced in Section 2 to give more insight into adjoint functors. The adjunction condition for a pair of adjoint functors $F \vdash G$

$$D(Fc, d) \simeq C(c, Gd)$$

implies that for any object $c \in C$, the object $Fc \in D$ represents the functor $C(c, G-) : D \to \mathbf{Set}$. Recall from the Yoneda Lemma that the natural isomorphism $D(Fc, -) \simeq C(c, G-)$ is determined by an element of $C(c, GFc)$, which can be viewed as the transpose of 1_{Fc}. Denoting such elements as η_c, they can be assembled jointly into the natural transformation $\eta : 1_C \to GF$. Below we will see that this forms one of the conditions for an endofunctor to define a monad.

Theorem 10. The **unit** $\eta : 1_C \to GF$ is a natural transformation defined by an adjunction $F \vdash G$, whose component $\eta_c : c \to GFc$ is defined to be the transpose of the identity morphism 1_{Fc}.

Proof: We need to show that for every $f : c \to c'$, the following diagram commutes, which follows from the definition of adjunction and the isomorphism condition that it imposes, as well as the obvious commutativity of the second transposed diagram below the first one.

$$\begin{array}{ccc} c & \xrightarrow{\eta_c} & GFc \\ \downarrow{\scriptstyle f} & & \downarrow{\scriptstyle GFf} \\ c' & \xrightarrow{\eta_{c'}} & GFc' \end{array}$$

$$\begin{array}{ccc} Fc & \xrightarrow{1_{Fc}} & Fc \\ \downarrow{\scriptstyle Ff} & & \downarrow{\scriptstyle Ff} \\ Fc' & \xrightarrow{1_{Fc'}} & Fc' \end{array}$$

The dual of the above theorem leads to the second major component of an adjunction.

Theorem 11. The **counit** $\epsilon : FG \Rightarrow 1_D$ is a natural transformation defined by an adjunction $F \vdash G$, whose components $\epsilon_c : FGd \to d$ at d is defined to be the transpose of the identity morphism 1_{Gd}.

Adjoint functors interact with universal constructions, such as limits and colimits, in ways that turn out to be important for a variety of applications in AI and ML. We state the main results here, but refer the reader to Riehl [2017] for detailed proofs. Before getting to the general case, it is illustrative to see the interaction of limits and colimits with adjoint functors for preorders. Recall from above that separoids are defined by a preorder $(S, \leqslant)$ on a join lattice of elements from a set S. Given two separoids $(S, \leqslant_S)$ and $(T, \leqslant_T)$, we can define the functors $F : S \to T$ and $G : T \to S$ to be order-preserving functions such that

$$Fa \leqslant_T b \quad \text{if and only if} \quad a \leqslant_S Gb$$

Such an adjunction between preorders is often called a *Galois connection*. For preorders, the limit is defined by the *meet* of the preorder, and the colimit is defined by the *join* of the preorder. We can now state a useful result. For a fuller discussion of preorders and their applications from a category theory perspective, see Fong and Spivak [2018].

Theorem 12. Right adjoints preserve meets in a preorder: Let $f : P \to Q$ be left adjoint to $g : Q \to P$, where P, Q are both preorders, and f and g are monotone order-preserving functions. For any subset $A \subseteq Q$, let $g(A) = \{g(a) | a \in Q\}$. If A has a meet $\bigwedge A \in Q$, then $g(A)$ has a meet $\wedge g(A) \in P$, and we can see that $g(\wedge A) \simeq \bigwedge g(A)$, that is, right adjoints preserve meets. Similarly, left adjoints preserve meets, so that if $A \subset P$ such that $\bigvee A \in P$ then $f(A)$ has a join $\vee f(A) \in Q$ and we can set $f(\vee A) \simeq \bigvee f(A)$, so that left adjoints preserve joins.

Proof: The proof is not difficult in this special case of the category being defined as a preorder. If $f : P \to Q$ and $g : Q \to P$ are monotone adjoint maps on preorders P, Q, and $A \subset Q$ is any subset such that its meet is $m = \wedge A$. Since g is monotone, $g(m) \leqslant g(a),\ \forall a \in A$, hence it follows that $g(m) \leqslant g(A)$. To show that $g(m$ is the greatest lower bound, if we take any other lower bound $b \leqslant g(a),\ \forall a \in A$, then we want to show that $b \leqslant g(m)$. Since f and g are adjoint, for every $p \in P, q \in Q$, we have

$$p \leqslant g(f(p)) \quad \text{and} \quad f(g(q)) \leqslant q$$

Hence, $f(b) \leqslant a$ for all $a \in A$, which implies $f(b)$ is a lower bound for A on Q. Since the meet m is the greatest lower bound, we have $f(b) \leqslant m$. Using the Galois connection, we see that $b \leqslant g(m)$, and hence showing that $g(m)$ is the greatest lower bound as required. An analogous proof follows to show that left adjoints preserve joins. □

We can now state the more general cases for any pair of adjoint functors, as follows.

Theorem 13. A category $\mathcal{C}$ admits all limits of diagrams indexed by a small category $\mathcal{J}$ if and only if the constant functor $\Delta : \mathcal{C} \to \mathcal{C}^{\mathcal{J}}$ admits a right adjoint, and admits all colimits of $\mathcal{J}$-indexed diagrams if and only if Δ admits a left adjoint.

By way of explanation, the constant functor $c : J \to C$ sends every object of J to c and every morphism of J to the identity morphism 1_c. Here, the constant functor Δ sends every object c of C to the constant diagram Δc, namely the functor that maps each object i of J to the object c and each morphism of J to the identity 1_c. The theorem follows from the definition of the universal properties of colimits and limits. Given any object $c \in C$, and any diagram (functor) $F \in \mathcal{C}^{\mathcal{J}}$, the set of morphisms $\mathcal{C}^{\mathcal{J}}(\Delta c, F)$ corresponds to the set of natural transformations from the constant $\mathcal{J}$-diagram at c to the diagram F. These natural transformations precisely correspond to the cones over F with summit c in the definition given earlier in Section 2. It follows that there is an object $\lim F \in \mathcal{C}$ together with an isomorphism

$$\mathcal{C}^{\mathcal{J}}(\Delta c, F) \simeq \mathcal{C}(c, \lim F)$$

We can now state the more general result that we showed above for the special case of adjoint functors on preorders.

Theorem 14. Right adjoints preserve limits, whereas left adjoints preserve colimits.

7 The Coend and End of GAIA: Integral Calculus for Generative AI

In this section, we introduce a powerful abstract integral calculus for generative AI based on the theory of coends and ends Yoneda [1960], Loregian [2021].

We build on two foundational results in category theory: the metric Yoneda Lemma Bonsangue et al. [1998] shows how to construct universal representations of generative AI models in generalized metric spaces where symmetry does not hold; and a categorical integral calculus also introduced by Yoneda Yoneda [1960] based on (co)ends, (initial) final objects in a category of (co)wedges. Loregian [2021] provides an excellent book-length treatment of Yoneda's categorical integral calculus of (co)ends. We define two classes of generative AI modes based on coends and ends. Coend generative AI models are defined by dinatural transformations between bifunctors $F : \mathcal{C}^{op} \times \mathcal{C} \to \mathcal{D}$ that combine a contravariant and covariant action. Here, $\mathcal{C}$ represents a generic category of generative AI models, modeled as a *twisted arrow* category. The co-domain category $\mathcal{D}$ is the category $\mathbf{Meas}$ of measurable spaces for generative AI models based on ends, and the category $\mathbf{Top}$ of topological spaces for the generative AI models based on coends. Recent theoretical results have shown that the traditional Transformer model is a universal approximator of sequences, despite the restriction of permutation equivariance, due to the use of absolute positional encoding of input tokens, which leads to poor generalization on long sequences. Modifications, such as relative positional encoding, impose limitations on the universal approximability of the traditional Transformer. We conjecture that coend generative AI models provide a non-symmetric measure of distance, and furthermore, capture higher-order interactions between tokens using the structure of simplicial sets.

Figure 26 illustrates the two fundamental insights developed by Yoneda that form the theoretical core of our GAIA framework. The celebrated Yoneda Lemma MacLane [1971] asserts that objects in a category $\mathcal{C}$ can be defined purely in terms of their interactions with other objects. This interaction is modeled by *contravariant* or *covariant* functors:

$$\mathcal{C}(-, x) : \mathcal{C}^{op} \to \mathbf{Sets}, \quad \mathcal{C}(x, -) : \mathcal{C} \to \mathbf{Sets}$$

The *Yoneda embedding* $x \to \mathcal{C}(-, x)$ is sometimes denoted as よ(x) for the Japanese Hiragana symbol for yo, serves as a *universal representer*, and generalizes many other similar ideas in machine learning, such as representers $K(-, x)$ in kernel methods Schölkopf and Smola [2002] and representers of causal information Mahadevan [2023]. There are many variants of the Yoneda Lemma, including versions that map the functors $\mathcal{C}(-, x)$ and $\mathcal{C}(x, -)$ into an *enriched* category. In particular, Bradley et al. [2022] contains an extended discussion of the use of an enriched Yoneda Lemma to model natural language interactions that result from using a large language model. In particular, we build on the metric Yoneda Lemma Bonsangue et al. [1998] that defines a universal representer in generalized metric spaces, where

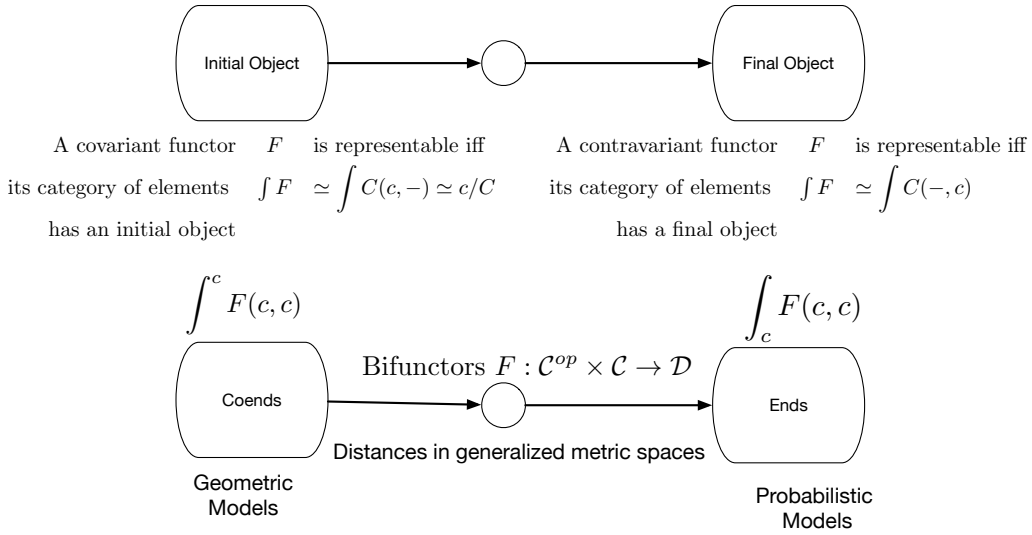


Figure 26: The theoretical foundation of GAIA is based on two celebrated results of Yoneda. The first (top row) shows that Yoneda embeddings $\yoneda(x) = C(-, x)$ are universal representers of objects in a category. We use this result to define universal representers of generative AI models. The second (bottom row) is based on Yoneda's categorical "integral calculus" using coends and ends Yoneda [1960], which defines two classes of generative AI models ranging from probabilistic models to topological models.

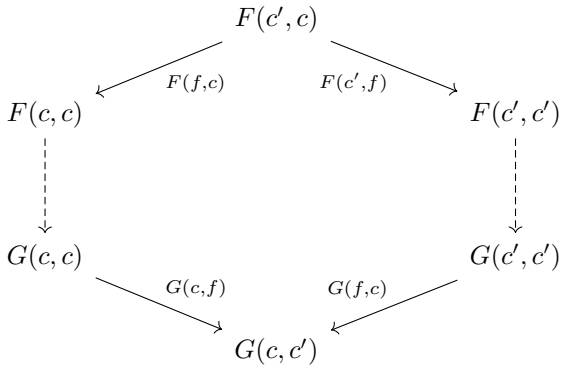
distances are non-symmetric. The second major insight from Yoneda Yoneda [1960] is based on a powerful concept of the *coend* and *end* of a *bifunctor* $F : \mathcal{C}^{op} \times \mathcal{C} \to \mathcal{D}$ that combines both a *contravariant* and a *covariant* action. We build on the insight that probabilistic generative models, or using distances in some metric space, correspond to final or initial objects in a category of wedges, defined by bifunctors, and the arrows are dinatural transformations. These initial or terminal objects correspond to coends and ends. Bifunctors $F : \mathcal{C}^{op} \times \mathcal{C} \to \mathcal{D}$ can be used to construct universal representers of distance functions in generalized metric spaces leading to a "metric Yoneda Lemma" Bonsangue et al. [1998].

Recent universal approximation results Yun et al. [2020] have shown that the category $\mathcal{C}_T$ of transformers is dense in the parent category of all permutation-equivariant functions on (compact) vector spaces $\mathcal{C}_{PE}$ defined by vectors $x \in \mathbb{R}^{n \times d}$ over arbitrary continuous permutation equivariant functions. We define a twisted arrow category $\mathcal{C}_{PE}^{TW}$, which has as its objects the equivariant maps of $\mathcal{C}_{PE}$, and commutative diagrams over pairs of equivariant maps f, g in $\mathcal{C}_{PE}$ as its morphisms. To define the (co)ends of Transformer models, we define a category of wedges defined by bifunctors $F : (\mathcal{C}_{PE}^{TW})^{op} \times \mathcal{C}_{PE}^{TW} \to \mathcal{D}$ that contravariantly and covariantly map Transformer models into $\mathcal{D}$, the codomain category, which may be the category **Meas** of measurable spaces, or the category of distances $[0, \infty]$ defined by l_p norms over permutation equivariant functions. We use the metric Yoneda Lemma to construct a *universal representer* of Transformer models in a generalized metric space. Building on Yoneda's categorical calculus of (co)ends, we define the end $\int_c F(c, c)$ of Transformer models as the final object in the category of wedges, whereas the coend $\int^c F(c, c)$ of Transformer modes are defined as the initial object in the category of cowedges, both defined over dinatural transformations between bifunctors over transformer models. Ends induce probabilistic generative models over sequences of tokens implemented as Transformer models, whereas coends lead to *Geometric Transformer Models* (GTMs), a new class of generative sequence models defined by the topological embedding of (fuzzy) simplicial sets.

7.1 Ends and Coends

We will analyze generative AI models in the category of *wedges*, which are defined by a collection of objects comprised of bifunctors $F : \mathcal{C}^{op} \times C \to \mathcal{D}$, and a collection of arrows between each pair of bifunctors F, G called a *dinatural transformation* (as an abbreviation for diagonal natural transformation). We will see below that the initial and terminal objects in the category of wedges correspond to a beautiful idea first articulated by Yoneda called the *coend* or *end* Yoneda [1960]. Loregian [2021] has an excellent treatment of coend calculus, which we will use below.

Definition 51. Given a pair of bifunctors $F, G : \mathcal{C}^{op} \times \mathcal{C} \to \mathcal{D}$, a **dinatural transformation** is defined as follows:



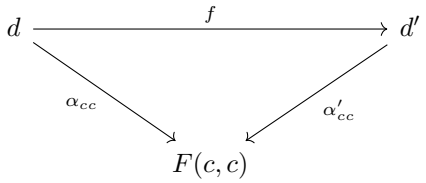
As Loregian [2021] observes, just as a natural transformation interpolates between two regular functors F and G by filling in the gap between their action on a morphism Ff and Fg on the codomain category, a dinatural transformation "fills in the gap" between the top of the hexagon above and the bottom of the hexagon.

We can define a *constant bifunctor* $\Delta_d : \mathcal{C}^{op} \times \mathcal{C} \to \mathcal{D}$ by the object it maps everything to, namely the input pair of objects $(c, c') \to d$ are both mapped to the object $d \in \mathcal{D}$, and the two input morphisms $(f, f') \to \mathbf{1}_d$ are both mapped to the identity morphism on d. We can now define *wedges* and *cowedges*.

Definition 52. A **wedge** for a bifunctor $F : \mathcal{C}^{op} \times \mathcal{C} \Rightarrow \mathcal{D}$ is a dinatural transformation $\Delta_d \to F$ from the constant functor on the object $d \in \mathcal{D}$ to F. Dually, we can define a **cowedge** for a bifunctor F by the dinatural transformation $P \Rightarrow \Delta_d$.

We can now define a *category of wedges*, each of whose objects are wedges, and for arrows, we choose arrows in the co-domain category that makes the diagram below commute.

Definition 53. Given a fixed bifunctor $F : \mathcal{C}^{op} \times \mathcal{C} \to \mathcal{D}$, we define the **category of wedges** $\mathcal{W}(F)$ where each object is a wedge $\Delta_d \Rightarrow F$ and given a pair of wedges $\Delta_d \Rightarrow F$ and $\Delta'_d \Rightarrow F$, we choose an arrow $f : d \to d'$ that makes the following diagram commute:



Analogously, we can define a **category of cowedges** where each object is defined as a cowedge $F \Rightarrow \Delta_d$.

With these definitions in place, we can once again define the universal property in terms of initial and terminal objects. In the category of wedges and cowedges, these have special significance for formulating and solving UIGs, as we will see in the next section.

Definition 54. Given a bifunctor $F : \mathcal{C}^{op} \times \mathcal{C} \to \mathcal{D}$, the **end** of F consists of a terminal wedge $\omega : \underline{\mathbf{end}}(F) \Rightarrow F$. The object $\underline{\mathbf{end}}(F) \in D$ is itself called the end. Dually, the **coend** of F is the initial object in the category of cowedges $F \Rightarrow \underline{\mathbf{coend}}(F)$, where the object $\underline{\mathbf{coend}}(F) \in \mathcal{D}$ is itself called the coend of F.

Remarkably, probabilities can be formally shown to define ends of a category Avery [2016], and topological embeddings of datasets, as implemented in popular dimensionality reduction methods like UMAP McInnes et al. [2018], correspond to coends MacLane [1971]. These connections suggest the canonical importance of the category of wedges and cowedges in formulating and solving UIGs. First, we introduce another universal construction, the Kan extension, which turns out to be the basis of every other concept in category theory.

7.2 Sheaves and Topoi in GAIA

So far, we have assumed that the parameter spaces for generative AI are vector spaces $\mathbb{R}^n$, as is typically assumed in deep learning Bengio [2009]. But there are excellent reasons to consider more abstract spaces, and in particular, we describe here an important category of sheaves and topoi MacLane and leke Moerdijk [1994] where some of the most interesting results in category theory, like the (metric) Yoneda Lemma, find their application.

In this section, we define an important categorical structure defined by sheaves and topoi MacLane and leke Moerdijk [1994]. Yoneda embeddings $よ(x) : \mathcal{C}^{op} \to \mathbf{Sets}$ define (pre)sheaves, which satisfy a number of crucial properties that make it remarkably similar to the category of **Sets**. The sheaf condition plays an important role in many applications of machine learning, from dimensionality reduction McInnes et al. [2018] to causal inference Mahadevan [2023]. MacLane and leke Moerdijk [1994] provides an excellent overview of sheaves and topoi, and how remarkably they unify much of mathematics, from geometry to logic and topology. We will give only the briefest of overviews here, and apply in the main ideas to the study of UIGs.

Figure 27: Two applications of sheaf theory in AI: (top) minimizing travel costs in weighted graphs satisfies the sheaf principle, one example of which is the Bellman optimality principle in dynamic programming Bertsekas [2005] and reinforcement learning Bertsekas [2019], Sutton and Barto [1998] (bottom): Approximating a function over a topological space must satisfy the sheaf condition.

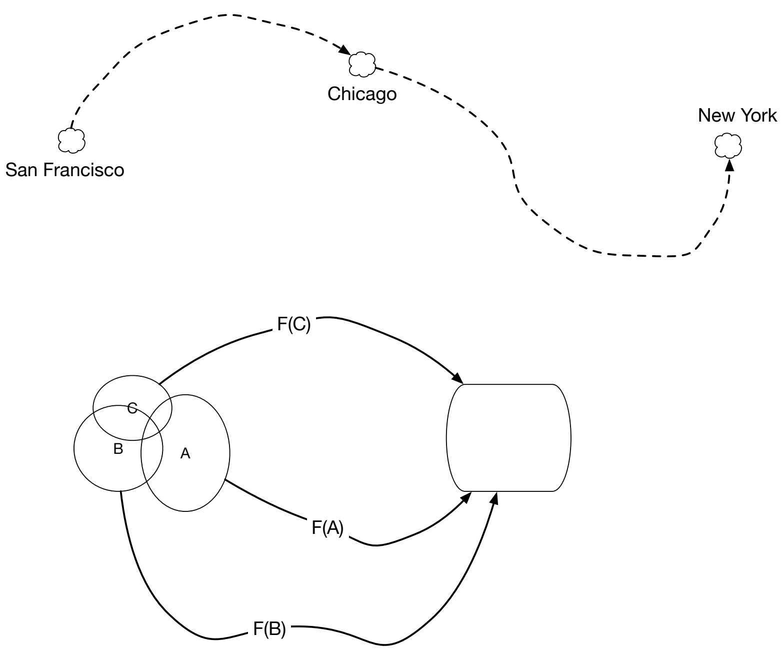


Figure 27 gives two concrete examples of sheaves. In a minimum cost transportation problem, say using optimal transport Villani [2003] or reinforcement learning Sutton and Barto [1998], any optimal solution has the property that any restriction of the solution must also be optimal. In RL, this sheaf principle is codified by the Bellman equation, and leads to the fundamental principle of dynamic programming Bertsekas [2005]. Consider routing candy bars from San Francisco to New York city. If the cheapest way to route candy bars is through Chicago, then the restriction of the overall route to the (sub) route from Chicago to New York City must also be optimal, otherwise it is possible to find a shortest overall route by switching to a lower cost route. Similarly, in function approximation with real-valued functions $F : \mathcal{C} \to \mathbb{R}$, where $\mathcal{C}$ is the category of topological spaces, the (sub)functions $F(A), F(B)$ and $F(C)$ restricted to the open sets A, B and C must agree on the values they map the elements in the intersections $A \cap B$, $A \cap C$, $A \cap B \cap C$ and so on. Similarly, in causal inference, any probability distribution that is defined over a causal generative AI model must satisfy the sheaf condition in that any restriction of the causal model to a submodel must be consistent, so that two causal submodels that overlap in their domains must agree on the common elements.

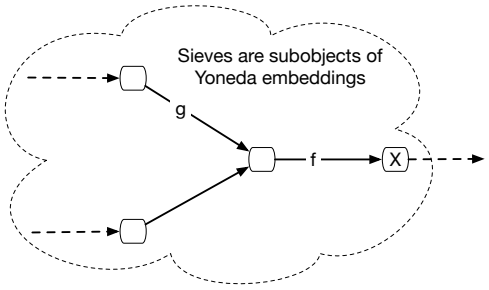
Sheaves can be defined over arbitrary categories, and we introduce the main idea by focusing on the category of sheaves over **Sets**.

Definition 55. MacLane and leke Moerdijk [1994] A **sheaf** of sets F on a topological space X is a functor $F : \mathcal{O}^{op} \to \mathbf{Sets}$ such that each open covering $U = \bigcup_i U_i, i \in I$ of an open set O of X yields an equalizer diagram

$$FU \overset{e}{\dashrightarrow} \prod_i FU_i \underset{q}{\overset{p}{\rightrightarrows}} \prod_{i,} F(U_i \cap U_j)$$

The above definition succinctly captures what Figure 27 shows for the example of approximating functions: the value of each subfunction must be consistent over the shared elements in the intersection of each open set.

Figure 28: Sieves are subobjects of of $\yoneda(x)$ Yoneda embeddings of a category $\mathcal{C}$, which generalizes the concept of sheaves over sets in Figure 27.



Definition 56. The category $\mathrm{Sh}(X)$ of sheaves over a space X is a full subcategory of the functor category $\mathbf{Sets}^{\mathcal{O}(X)^{op}}$.

Grothendieck Topologies

We can generalize the notion of sheaves to arbitrary categories using the Yoneda embedding $\yoneda(x) = \mathcal{C}(-, x)$. We explain this generalization in the context of a more abstract topology on categories called the *Grothendieck topology* defined by *sieves*. A sieve can be viewed as a *subobject* $S \subseteq \yoneda(x)$ in the presheaf $\mathbf{Sets}^{\mathcal{C}^{op}}$, but we can define it more elegantly as a family of morphisms in $\mathcal{C}$, all with codomain x such that

$$f \in S \Longrightarrow f \circ g \in S$$

Figure 28 illustrates the idea of sieves. A simple way to think of a sieve is as a *right ideal*. We can define that more formally as follows:

Definition 57. If S is a sieve on x, and $h : D \to x$ is any arrow in category $\mathcal{C}$, then

$$h^* = \{g \mid \mathrm{cod}(g) = D, hg \in S\}$$

Definition 58. MacLane and leke Moerdijk [1994] A **Grothendieck topology** on a category $\mathcal{C}$ is a function J which assigns to each object x of $\mathcal{C}$ a collection $J(x)$ of sieves on x such that

1. the maximum sieve $t_x = \{f | \mathrm{cod}(f) = x\}$ is in $J(x)$.
2. If $S \in J(x)$ then $h^*(S) \in J(D)$ for any arrow $h : D \to x$.
3. If $S \in J(x)$ and R is any sieve on x, such that $h^*(R) \in J(D)$ for all $h : D \to x$, then $R \in J(C)$.

We can now define categories with a given Grothendieck topology as *sites*.

Definition 59. A **site** is defined as a pair $(\mathcal{C}, J)$ consisting of a small category $\mathcal{C}$ and a Grothendieck topology J on $\mathcal{C}$.

An intuitive way to interpret a site is as a generalization of the notion of a topology on a space X, which is defined as a set X together with a collection of open sets $\mathcal{O}(X)$. The sieves on a category play the role of "open sets".

Exponential Objects and Cartesian Closed Categories

To define a topos, we need to understand the category of **Sets** a bit more. Clearly, the single point set $\{\bullet\}$ is a terminal object for **Sets**, and the binary product of two sets $A \times B$ can always be defined. Furthermore, given two sets A and B, we can define B^A as the exponential object representing the set of all functions $f : A \to B$. We can define exponential objects in any category more generally as follows.

Definition 60. Given any category $\mathcal{C}$ with products, for a fixed object x in $\mathcal{C}$, we can define the functor

$$x \times - :\to \mathcal{C}$$

If this functor has a right adjoint, which can be denoted as

$$(-)^x : \mathcal{C} \to \mathcal{C}$$

then we say x is an **exponentiable** object of $\mathcal{C}$.

Definition 61. A category $\mathcal{C}$ is **Cartesian closed** if it has finite products (which is equivalent to saying it has a terminal object and binary products) and if all objects in $\mathcal{C}$ are *exponentiable*.

A result that is of foundational importance to this paper is that the category defined by Yoneda embeddings is Cartesian closed.

Theorem 15. MacLane and leke Moerdijk [1994] For any small category $\mathcal{C}$, the functor category $\mathbf{Sets}^{\mathcal{C}^{op}}$ is Cartesian closed

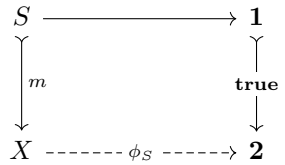
For a detailed proof, the reader is referred to MacLane and leke Moerdijk [1994]. A further result of significance is the *density theorem*, which can be seen as the generalization of the simple result that any set S can be defined as the union of single point sets $\bigcup_{x \in S}\{x\}$.

Theorem 16. MacLane and leke Moerdijk [1994] In a functor category $\mathbf{Sets}^{\mathcal{C}^{op}}$, any object x is the colimit of a diagram of representable objects in a canonical way.

Recall that an object is representable if it is isomorphic to a Yoneda embedding $\yoneda(x)$. This result has numerous applications to AI and ML, among them to causal inference Mahadevan [2023] and universal decision models Mahadevan [2021b].

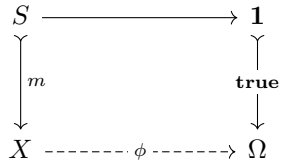
Subobject Classifiers

A topos builds on the property of subobject classifiers in **Sets**. Given any subset $S \subset X$, we can define S as the monic arrow $S \hookrightarrow X$ defined by the inclusion of S in X, or as the characteristic function ϕ_S that is equal to 1 for all elements $x \in X$ that belong to S, and takes the value 0 otherwise. We can define the set $\mathbf{2} = \{0, 1\}$ and treat **true** as the inclusion $\{1\}$ in $\mathbf{2}$. The characteristic function ϕ_S can then be defined as the pullback of **true** along ϕ_S.



We can now define subobject classifiers in a category $\mathcal{C}$ as follows.

Definition 62. In a category $\mathcal{C}$ with finite limits, a **subobject classifier** is a *monic* arrow $\mathbf{true} : \mathbf{1} \to \Omega$, such that to every other monic arrow $S \hookrightarrow X$ in $\mathcal{C}$, there is a unique arrow ϕ that forms the following pullback square:



This definition can be rephrased as saying that the subobject functor is representable. In other words, a subobject of an object x in a category $\mathcal{C}$ is an equivalence class of monic arrows $m : S \hookrightarrow x$.

MacLane and leke Moerdijk [1994] provide many examples of subobject classifiers. Vigna [2003] gives a detailed description of the topos of graphs.

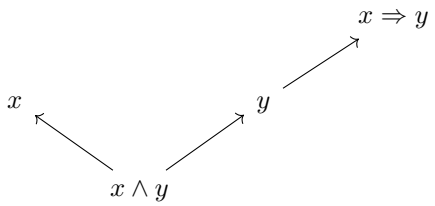
Heyting Algebras

A truly remarkable finding is that the logic of topoi is not classical Boolean logic, but intuitionistic logic defined by *Heyting algebras*.

Definition 63. A **Heyting algebra** is a poset with all finite products and coproducts, which is Cartesian closed. That is, a Heyting algebra is a lattice with $\mathbf{0}$ and $\mathbf{1}$ which has to each pair of elements x and y an exponential y^x. The exponential is written $x \Rightarrow y$, and defined as the adjunction

$$z \leqslant (x \Rightarrow y) \text{ if and only if } z \wedge x \leqslant y$$

Alternatively, $x \Rightarrow y$ is a least upper bound for all those elements z with $z \wedge x \leqslant y$. Therefore, for the particular case of y, we get that $y \leqslant (x \Rightarrow y)$. In the figure below, the arrows show the partial ordering relationship. As a concrete example, for a topological space X the set of open setx $\mathcal{O}(X)$ is a Heyting algebra. The binary intersections and unions of open sets yield open sets. The empty set $\emptyset$ represents $\mathbf{0}$ and the complete set X represents $\mathbf{1}$. Given any two open sets U and V, the exponential object $U \Rightarrow W$ is defined as the union $\bigcup_i W_i$ of all open sets W_i for which $W \cap U \subset V$.



Note that in a Boolean algebra, we define implication as the relationship

$$(x \Rightarrow y) \equiv \neg x \vee y$$

This property, which is sometimes referred to as the "law of the excluded middle" (because if $x = y$, then this translates to $\neg x \vee x = \mathbf{true}$), does not hold in a Heyting algebra. For example, on a real line $\mathbb{R}$, if we define the open sets by the open intervals $(a, b), a, b \in \mathbb{R}$, the complement of an open set need not be open.

We can now state what is a truly remarkable result about the subobjects of a (pre)sheaf.

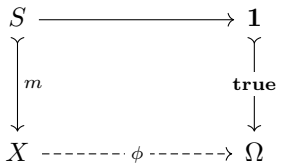
Theorem 17. MacLane and leke Moerdijk [1994] For any functor category $\hat{C} = \mathbf{Sets}^{\mathcal{C}^{op}}$ of a small category $\mathcal{C}$, the partially ordered set $\text{Sub}_{\hat{C}}(x)$ of subobjects of x, for any object x of $\hat{C}$ is a Heyting algebra.

This result has deep implications for a lot of applications in AI and ML that are based modeling presheaves, including causal inference and decision making. It implies that the proper logic to employ in these settings is intuitionistic logic, not classical logic as is often used in AI Pearl [2009], Fagin et al. [1995], Halpern [2016].

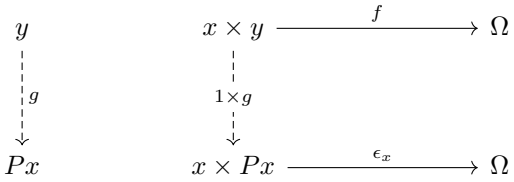
Finally, we can now define the category of topoi.

Definition 64. A **topos** is a category $\mathcal{E}$ with

1. A pullback for every diagram $X \to B \leftarrow Y$.
2. A terminal object $\mathbf{1}$.
3. An object Ω and a monic arrow $\mathbf{true} : 1 \to \Omega$ such that any monic $m : S \hookrightarrow B$, there is a unique arrow $\phi : B \to \Omega$ in $\mathcal{E}$ for which the following square is a pullback:



4. To each object x an object Px and an arrow $\epsilon_x : x \times Px \to \Omega$ such that for every arrow $f : x \times y \to \Omega$, there is a unique arrow $g : y \to Px$ for which the following diagrams commute:



7.3 Topological Embedding of Simplicial Sets

Simplicial sets can be embedded in a topological space using coends MacLane [1971], which is the basis for a popular machine learning method for reducing the dimensionality of data called UMAP (Uniform Manifold Approximation and Projection) McInnes et al. [2018].

Definition 65. The **geometric realization** $|X|$ of a simplicial set X is defined as the topological space

$$|X| = \bigsqcup_{n \geqslant 0} X_n \times \Delta^n / \sim$$

where the n-simplex X_n is assumed to have a *discrete* topology (i.e., all subsets of X_n are open sets), and Δ^n denotes the *topological* n-simplex

$$\Delta^n = \{(p_0, \ldots, p_n) \in \mathbb{R}^{n+1} \mid 0 \leqslant p_i \leqslant 1, \sum_i p_i = 1$$

The spaces $\Delta^n, n \geqslant 0$ can be viewed as *cosimplicial* topological spaces with the following degeneracy and face maps:

$$\delta_i(t_0, \ldots, t_n) = (t_0, \ldots, t_{i-1}, 0, t_i, \ldots, t_n) \text{ for } 0 \leqslant i \leqslant n$$

$$\sigma_j(t_0, \ldots, t_n) = (t_0, \ldots, t_j + t_{j+1}, \ldots, t_n) \text{ for } 0 \leqslant i \leqslant n$$

Note that $\delta_i : \mathbb{R}^n \to \mathbb{R}^{n+1}$, whereas $\sigma_j : \mathbb{R}^n \to \mathbb{R}^{n-1}$.

The equivalence relation $\sim$ above that defines the quotient space is given as:

$$(d_i(x), (t_0, \ldots, t_n)) \sim (x, \delta_i(t_0, \ldots, t_n))$$

$$(s_j(x), (t_0, \ldots, t_n)) \sim (x, \sigma_j(t_0, \ldots, t_n))$$

Topological Embeddings as Coends

We now bring in the perspective that topological embeddings can be interpreted as coends as well. Consider the functor

$$F : \Delta^o \times \Delta \to \text{Top}$$

where

$$F([n], [m]) = X_n \times \Delta^m$$

where F acts *contravariantly* as a functor from Δ to $\mathbf{Sets}$ mapping $[n] \mapsto X_n$, and *covariantly* mapping $[m] \mapsto \Delta^m$ as a functor from Δ to the category Top of topological spaces.

7.4 The Geometric Transformer Model

In this section, we define the Geometric Transformer Model (GTM), which arises as a coend object defined by the topological embedding of a simplicial set defined over n-length sequences of tokens of dimension d. Given the restrictions on space, we can only give a very brief explanation, and a more detailed analysis is the topic of a future paper.

Given a category of generative AI models, such as Transformers defined as permutation equivariant functions over $\mathbb{R}^{d \times n}$, it is possible to construct simplicial sets by constructing the *nerve* of the category. So, for example, the nerve of the category $\mathcal{C}_T$ of Transformers is a simplicial set Transformer$_\bullet$, comprised of a sequence of composable morphisms of length $n \geqslant 0$, each defining a Transformer block. Given this simplicial set, we can now construct a topological realization of it as a coend object

$$\int^{n} (\text{Transformer}_\bullet n) \cdot \Delta n$$

where Transformer$_\bullet : \Delta^{op} \to \mathcal{C}_T$ is a contravariant functor from the simplicial category Δ into the category of Transformers, and $\Delta : |\Delta| \to \mathbf{Top}$ is a functor from the topological n-simplex realization of the simplicial category Δ into topological spaces $\mathbf{Top}$. As MacLane [1971] explains it picturesquely, the "coend formula describes the geometric realization in one gulp". The formula says essentially to take the disjoint union of affine n-simplices, one for each $t \in \text{Transformers}_\bullet n$, and glue them together using the face and degeneracy operations defined as arrows of the simplicial category Δ. In more concrete terms, this coend formula is essentially what the UMAP method implements for point cloud data in Euclidean space. Here, we are generalizing this application to construct topological realization of generative AI models, such as Transformers.

7.5 The End of GAIA: Monads and Categorical Probability

We now turn to discuss the ends of generative AI models, where we first show that categorically speaking, probabilities are defined as end objects Avery [2016]. This notion requires defining particular tyoes of functors called monads more formally, and relate them to adjoint functors. Categorically speaking, probabilities are essentially monads Avery [2016]. Like the case with coalgebras, which we discussed extensively in previous Sections, monads also are defined by an endofunctor on a category, but one that has some special properties. These additional properties make monads possess algebraic structure, which leads to many interesting properties. Monads provide a categorical foundation for probability, based on the property that the set of all distributions on a measurable space is itself a measurable space. The well-known *Giry* monad been also shown to arise as the *codensity monad* of a forgetful functor from the category of convex sets with affine maps to the category of measurable spaces Avery [2016]. Our goal in this paper is to apply monads to shed light into causal inference. We first review the basic definitions of monads, and then discuss monad algebras, which provide ways of characterizing categories.

Consider the pair of adjoint free and forgetful functors between graphs and categories. Here, the domain category is **Cat**, the category of all categories whose objects are categories and whose morphisms are functors. The co-domain category is the category **Graph** of all graphs, whose objects are directed graphs, and whose morphisms are graph homomorphisms. Here, a monad $T = U \circ F$ is induced by composing the "free" functor F that maps a graph into its associated "free" category, and the "forgetful" functor U that maps a category into its associated graph. The monad T in effect takes a directed graph G and computes its transitive closure G_{tc}. More precisely, for every (directed) graph G, there is a universal arrow from G to the "forgetful" functor U mapping the category **Cat** of all categories to **Graph**, the category of all (directed) graphs, where for any category C, its associated graph is defined by $U(C)$.

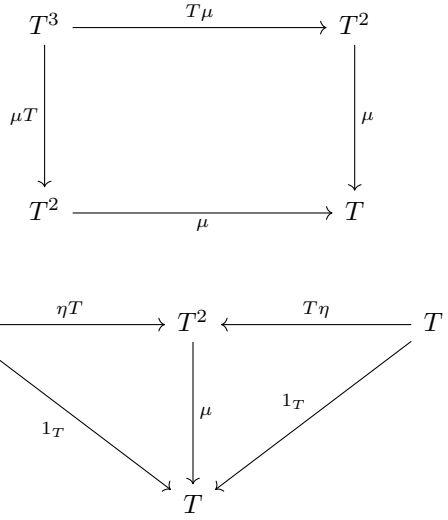
To understand this functor, simply consider a directed graph $U(C)$ as a category C forgetting the rule for composition. That is, from the category C, which associates to each pair of composable arrows f and g, the composed arrow $g \circ f$, we derive the underlying graph $U(G)$ simply by forgetting which edges correspond to elementary functions, such as f or g, and which are composites. The universal arrow from a graph G to the forgetful functor U is defined as a pair $\langle G, u : G \to U(C) \rangle$, where u is a a graph homomorphism. This arrow possesses the following *universal property*: for every other pair $\langle D, v : G \to H \rangle$, where D is a category, and v is an arbitrary graph homomorphism, there is a functor $f' : C \to D$, which is an arrow in the category **Cat** of all categories, such that *every* graph homomorphism $\phi : G \to H$ uniquely factors through the universal graph homomorphism $u : G \to U(C)$ as the solution to the equation $\phi = U(f') \circ u$, where $U(f') : U(C) \to H$ (that is, $H = U(D)$). Namely, the dotted arrow defines a graph homomorphism $U(f')$ that makes the triangle diagram "commute", and the associated "extension" problem of finding this new graph homomorphism $U(f')$ is solved by "lifting" the associated category arrow $f' : C \to D$. In causal inference using graph-based models, the transitive closure graph is quite important in a number of situations. It can be the initial target of a causal discovery algorithm that uses conditional independence oracles. It is also common in

graph-based causal inference Pearl [2009] to model causal effects through a directed acyclic graph (DAG) G, which specifies its algebraic structure, and through a set of probability distributions on G that specifies its semantics $P(G)$. Often, reasoning about causality in a DAG requires examining paths that lead from some vertex x, representing a causal variable, to some other vertex y. The process of constructing the transitive closure of a DAG provides a simple example of a causal monad.

Definition 66. A **monad** on a category C consists of

- An endofunctor $T : C \to C$
- A **unit** natural transformation $\eta : 1_C \Rightarrow T$
- A **multiplication** natural transformation $\mu : T^2 \to T$

such that the following commutative diagram in the category C^C commutes (notice the arrows in this diagram are natural transformations as each object in the diagram is a functor).



It is useful to think of monads as the "shadow" cast by an adjunction on the category corresponding to the co-domain of the right adjoint G. Consider the following pair of adjoint functors $F \vdash G$.

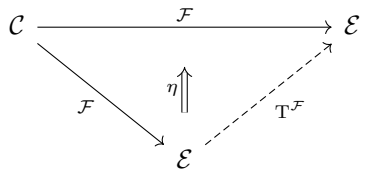
$$\mathcal{C} \underset{G}{\overset{F}{\underset{\longleftarrow}{\overset{\longrightarrow}{\bot}}}} \mathcal{D}. \qquad \eta : 1_C \Rightarrow UF, \quad \epsilon : FU \Rightarrow 1_D$$

In the language of ML, if we treat category C as representing "labeled training data" where we have full information, and category D as representing a new domain for which we have no labels, what can we conclude about category D from the information we have from the adjunction? The endofunctor UF on C is of course available to us, as is the natural transformation $\eta : 1_C \Rightarrow UF$. The map $\epsilon_A : FGA \to A$ for any object $A \in D$ is an endofunctor on D, about which we have no information. However, the augmented natural transformation $U\epsilon FA : GFGFA \to GFA$ can be studied in category C. From this data, what can we conclude about the objects in category D? In response to the natural question of whether every monad can be defined by a pair of adjoint functors, two solutions arose that came about from two different pairs of adjoint functors. These are referred to as the *Eilenberg-Moore* category and the *Kleisli* category MacLane [1971].

Codensity Monads and Probability

A striking recent finding is that categorical probability structures, such as Giry monads, are in essence *codensity monads* that result from extending a certain functor along itself Avery [2016].

Definition 67. A **codensity monad** $T^{\mathcal{F}}$ of a functor $\mathcal{F}$ is the right Kan extension of $\mathcal{F}$ along itself (if it exists). The codensity monad inherits the university property from the Kan extension.



Codensity monads can also be written using Yoneda's abstract integral calculus as ends:

$$T^{\mathcal{F}}e = \int_{c \in C} [\mathcal{E}(e, \mathcal{F}c), \mathcal{F}c]$$

Here, the notation $[A, m]$, where A is any set, and m is any object of a category $\mathcal{M}$, denotes the product in $\mathcal{M}$ of A copies of m.

Definition 68. A **convex set** c is a convex subset of a real vector space, where for all $x, y \in c$, and for all $r \in [0, 1]$, the convex combination $rx + (1 - r)y \in c$. An **affine** map $h : c \to c'$ is a function such that $h(x +_r y) = h(x) +_r h(y)$ where $x +_r y = rx + (1 - r)y, r \in [0, 1]$.

To define categorical probability as codensity monads, we need to define forgetful functors from the category $\mathcal{C}'$ of compact convex subsets of $\mathbb{R}^n$ with affine maps to the category $\mathbf{Meas}$ of measurable spaces and measurable functions. In addition, let $\mathcal{D}'$ be the category $\mathcal{C}'$ with the object d_0 adjoined, where d_0 is the convex set of convergent sequences in the unit interval $I = [0, 1]$.

Theorem 18. Avery [2016] The $\mathcal{C}'$ be the category of compact convex subsets of $\mathbb{R}^n$ for varying n with affine maps between them, and let $\mathcal{D}'$ be the same with the object d_0 adjoined. Then, the codensity monads of the forgetful functors $U' : \mathcal{C}' \to \mathbf{Meas}$ and $V' : \mathcal{D}' \to \mathbf{Meas}$ are the finitely additive Giry monad and the Giry monad respectively.

The well-known Giry monad defines probabilities in both the discrete case and the continuous case (over Polish spaces) in terms of endofunctor on the category of measurable spaces. We can view Transformers as essentially defining a Giri monad over the space of all sequences of tokens representing strings of words in natural language. In effect, Transformers are an end, and there is much more to be described here than we have space in this paper. The complete analysis of the ends of GAIA models is the topic of a subsequent paper.

8 Homotopy and Classifying Spaces of Generative AI Models

In this section, we introduce the concept of a *classifying space* of a generative AI model, such as a Transformer network or a stable diffusion step or a structured state space sequence model. Each of these define composable morphisms. The sequence of such composable morphisms defines a simplicial set through the nerve functor, and the classifying space corresponds to the topological realization of the simplicial set. This construction of a classifying space is an example of homotopy theory in categories Richter [2020], which gives us ways to abstractly compare generative AI models.

8.1 Homotopy in Categories

To motivate the need to consider *homotopical equivalence*, we consider the following problem: a generative AI system can be used to construct summaries of documents, which raises the question of how to decide if a document summary reflects the actual document. If we view a document as an object in a category, then the question becomes one of deciding object equivalence in a looser sense of homotopy, namely is there an invertible transformation between the original document and its summary? We discuss how to construct the topological embedding of an arbitrary category by embedding it into a simplicial set by constructing its nerve, and then finding the topological embedding of the nerve using the *homotopy colimit* Richter [2020]. First, we discuss the topological embedding of a simplicial set, and formulate it in terms of computing a coend. As another example, causal generative AI models can only be determined up to some equivalence class from data, and while many causal discovery algorithms assume arbitrary interventions can be carried out to discover the unique structure, such interventions are generally impossible to do in practical applications. The concept of *essential graph* Andersson et al. [1997] is based on defining a "quotient space" of graphs, but similar issues arise more generally for non-graph based models as well. Thus, it is useful to understand how to formulate the notion of equivalent classes of causal generative AI models in an arbitrary category. For example, given the conditional independence structure $A \perp\!\!\!\perp B|C$, there are at least three different symmetric monoidal categorical representations that all satisfy this conditional independence Fong [2012], Jacobs et al. [2019], Fritz and Klingler [2023], and we need to define the quotient space over all such equivalent categories.

In our previous work on causal homotopy Mahadevan [2021a], we exploited the connection between causal DAG graphical models and finite topological spaces. In particular, for a DAG model $G = (V, E)$, it is possible to define a finite space topology $\mathcal{T} = (V, \mathcal{O})$, whose open sets $\mathcal{O}$ are subsets of the vertices V such that each vertex x is associated with an open set U_x defined as the intersection of all open sets that contain x. This structure is referred to an *Alexandroff* topology, which can be shown to emerge from universal representers defined by Yoneda embeddings in generalized metric spaces. An intuitive way to construct an Alexandroff topology is to define the open set for each variable x by the set of its ancestors A_x, or by the set of its descendants D_x. This approach transcribes a DAG graph into a finite topological space, upon which the mathematical tools of algebraic topology can be applied to construct homotopies among equivalent causal generative AI models. Our approach below generalizes this construction to simplicial objects, as well as general categories.

8.2 The Category of Fractions: Localizing Invertible Morphisms in a Generative AI Category

One way to pose the question of homotopy is to ask whether a category can be reduced in some way such that all invertible morphisms can be "localized" in some way. The problem of defining a category with a given subclass of invertible morphisms, called the category of fractions [Gabriel et al., 1967], is another concrete illustration of the close relationships between categories and graphs. Borceux [1994] has a detailed discussion of the "calculus of fractions", namely how to define a category where a subclass of morphisms are to be treated as isomorphisms. The formal definition is as follows:

Definition 69. Consider a category $\mathcal{C}$ and a class Σ of arrows of $\mathcal{C}$. The **category of fractions** $\mathcal{C}(\Sigma^{-1})$ is said to exist when a category $\mathcal{C}(\Sigma^{-1})$ and a functor $\phi : \mathcal{C} \to \mathcal{C}(\Sigma^{-1})$ can be found with the following properties:

1. $\forall f, \phi(f)$ is an isomorphism.
2. If $\mathcal{D}$ is a category, and $F : \mathcal{C} \to \mathcal{D}$ is a functor such that for all morphisms $f \in \Sigma$, $F(f)$ is an isomorphism, then there exists a unique functor $G : \mathcal{C}(\Sigma^{-1}) \to \mathcal{D}$ such that $G \circ \phi = F$.

A detailed construction of the category of fractions is given in Borceux [1994], which uses the underlying directed graph skeleton associated with the category.

8.3 Homotopy of Simplicial Generative AI Objects

We will discuss homotopy in categories more generally now. This notion of homotopy generalizes the notion of homotopy in topology, which defines why an object like a coffee cup is topologically homotopic to a doughnut (they have the same number of "holes").

Definition 70. Let C and C' be a pair of objects in a category $\mathcal{C}$. We say C is **a retract** of C' if there exists maps $i : C \to C'$ and $r : C' \to C$ such that $r \circ i = \mathrm{id}_{\mathcal{C}}$.

Definition 71. Let $\mathcal{C}$ be a category. We say a morphism $f : C \to D$ is a **retract of another morphism** $f' : C \to D$ if it is a retract of f' when viewed as an object of the functor category $\mathbf{Hom}([1], \mathcal{C})$. A collection of morphisms T of $\mathcal{C}$ is **closed under retracts** if for every pair of morphisms f, f' of $\mathcal{C}$, if f is a retract of f', and f' is in T, then f is also in T.

Definition 72. Let X and Y be simplicial sets, and suppose we are given a pair of morphisms $f_0, f_1 : X \to Y$. A **homotopy** from f_0 to f_1 is a morphism $h : \Delta^1 \times X \to Y$ satisfying $f_0 = h|_{0 \times X}$ and $f_1 = h_{1 \times X}$.

Classifying Spaces and Homotopy Colimits of Generative AI Models

Building on the intuition proposed above, we now introduce a construction of a topological space associated with the nerve of a category. As we saw above, the nerve of a category is a full and faithful embedding of a category as a simplicial object.

Definition 73. The **classifying space** of a category $\mathcal{C}$ is the topological space associated with the nerve of the category $|N_\bullet \mathcal{C}|$

To understand the classifying space $|N_\bullet \mathcal{C}|$ of a category $\mathcal{C}$, let us go over some simple examples to gain some insight.

Example 17. For any set X, which can be defined as a discrete category $\mathcal{C}_X$ with no non-trivial morphisms, the classifying space $|N_\bullet \mathcal{C}_X|$ is just the discrete topology over X (where the open sets are all possible subsets of X).

Example 18. If we take a partially ordered set $[n]$, with its usual order-preserving morphisms, then the nerve of $[n]$ is isomorphic to the representable functor $\delta(-, [n])$, as shown by the Yoneda Lemma, and in that case, the classifying space is just the topological space Δ_n defined above.

Definition 74. The **homotopy colimit** of the nerve of the category of elements associated with the set-valued functor $\delta : \mathcal{C} \rightarrow$ **Set** mapping the category $\mathcal{C}$ into the category of **Sets**, namely $N_\bullet \left(\int \delta \right)$.

We can extend the above definition straightforwardly to these cases using an appropriate functor $\mathcal{T}$: **Set** $\rightarrow$ **Top**, or alternatively $\mathcal{M}$: **Set** $\rightarrow$ **Meas**. These augmented constructions can then be defined with respect to a more general notion called the *homotopy colimit* Richter [2020] of a generative AI model.

Definition 75. The **topological homotopy colimit** $\text{hocolim}_{\mathcal{T} \circ \delta}$ of a category $\mathcal{C}$, along with its associated category of elements associated with a set-valued functor $\delta : \mathcal{C} \rightarrow$ **Set**, and a topological functor $\mathcal{T}$: **Set** $\rightarrow$ **Top** is isomorphic to topological space associated with the nerve of the category of elements, that is $\text{hocolim}_{\mathcal{T} \circ \delta} \simeq |N_\bullet \left(\int \delta \right)|$.

8.4 The Singular Homology of a Generative AI Model

Our goal is to define an abstract notion of an object in terms of its underlying classifying space as a category, and show how it can be useful in defining homotopy. We will also clarify how it relates to determining equivalences among objects, namely homotopical invariance, and also how it sheds light on UIGs. We build on the topological realization of n-simplices defined above. Define the set of all morphisms $\text{Sing}_n(X) = \mathbf{Hom}_{\mathbf{Top}}(\Delta_n, |\mathcal{N}_\bullet(\mathcal{C})|)$ as the set of singular n-simplices of $|\mathcal{N}_\bullet(\mathcal{C})|$.

Definition 76. For any topological space defined by $|\mathcal{N}_\bullet(\mathcal{C})|$, the **singular homology groups** $H_*(|\mathcal{N}_\bullet(\mathcal{C})|; \mathbf{Z})$ are defined as the homology groups of a chain complex

$$\ldots \xrightarrow{\partial} \mathbf{Z}(\text{Sing}_2(|\mathcal{N}_\bullet(\mathcal{C})|)) \xrightarrow{\partial} \mathbf{Z}(\text{Sing}_1(|\mathcal{N}_\bullet(\mathcal{C})|)) \xrightarrow{\partial} \mathbf{Z}(\text{Sing}_0(|\mathcal{N}_\bullet(\mathcal{C})|))$$

where $\mathbf{Z}(\text{Sing}_n(|\mathcal{N}_\bullet(\mathcal{C})|))$ denotes the free Abelian group generated by the set $\text{Sing}_n(|\mathcal{N}_\bullet(\mathcal{C})|)$ and the differential ∂ is defined on the generators by the formula

$$\partial(\sigma) = \sum_{i=0}^{n} (-1)^i d_i \sigma$$

Intuitively, a chain complex builds a sequence of vector spaces that can be used to construct an algebraic invariant of a generative AI model from its classifying space by choosing the left **k** module **Z** to be a vector space. Each differential ∂ then becomes a linear transformation whose representation is constructed by modeling its effect on the basis elements in each $\mathbf{Z}(\text{Sing}_n(X))$.

Example 19. Let us illustrate the singular homology groups defined by an integer-valued multiset Studeny [2010] used to model conditional independence. Imsets over a DAG of three variables $N = \{a, b, c\}$ can be viewed as a finite discrete topological space. For this topological space X, the singular homology groups $H_*(X; \mathbf{Z})$ are defined as the homology groups of a chain complex

$$\mathbf{Z}(\text{Sing}_3(X)) \xrightarrow{\partial} \mathbf{Z}(\text{Sing}_2(X)) \xrightarrow{\partial} \mathbf{Z}(\text{Sing}_1(X)) \xrightarrow{\partial} \mathbf{Z}(\text{Sing}_0(X))$$

where $\mathbf{Z}(\text{Sing}_i(X))$ denotes the free Abelian group generated by the set $\text{Sing}_i(X)$ and the differential ∂ is defined on the generators by the formula

$$\partial(\sigma) = \sum_{i=0}^{4} (-1)^i d_i \sigma$$

The set $\text{Sing}_n(X)$ is the set of all morphisms $\mathbf{Hom}_{Top}(|\Delta_n|, X)$. For an imset over the three variables $N = \{a, b, c\}$, we can define the singular n-simplex σ as:

$$\sigma : |\Delta^4| \rightarrow X \text{ where } |\Delta^n| = \{t_0, t_1, t_2, t_3 \in [0, 1]^4 : t_0 + t_1 + t_2 + t_3 = 1\}$$

The n-simplex σ has a collection of faces denoted as $d_0\sigma, d_1\sigma, d_2\sigma$ and $d_3\sigma$. If we pick the k-left module $\mathbf{Z}$ as the vector space over real numbers $\mathbb{R}$, then the above chain complex represents a sequence of vector spaces that can be used to construct an algebraic invariant of a topological space defined by the integer-valued multiset. Each differential ∂ then becomes a linear transformation whose representation is constructed by modeling its effect on the basis elements in each $\mathbf{Z}(\text{Sing}_n(X))$. An alternate approach to constructing a chain homology for an integer-valued multiset is to use Möbius inversion to define the chain complex in terms of the nerve of a category (see our recent work on categoroids [Mahadevan, 2022] for details).

9 Summary and Future Work

In this paper, we proposed a theoretical blueprint for a “next-generation" Generative AI Architecture (GAIA) that potentially lie beyond the scope of what is achievable with compositional learning methods such as backpropagation, the longstanding algorithmic workhorse of deep learning. Backpropagation can be conceptualized as a sequence of modules, where each module updates its parameters based on information it receives from downstream modules, and in turn, transmits information back to upstream modules to guide their updates. GAIA is based on a fundamentally different *hierarchical model*. Modules in GAIA are organized into a simplicial complex, much like business units in a company. Each n-simplicial complex acts like a manager: it receives updates from its superiors and transmits information back to its $n+1$ subsimplicial complexes that are its subordinates. To ensure this simplicial generative AI organization behaves coherently, GAIA builds on the mathematics of the higher-order category theory of simplicial sets and objects. Computations in GAIA, from query answering to foundation model building, are posed in terms of lifting diagrams over simplicial objects. The problem of machine learning in GAIA is modeled as “horn" extensions of simplicial sets: each sub-simplicial complex tries to update its parameters in such a way that a lifting diagram is solved. Traditional approaches used in generative AI using backpropagation can be used to solve “inner" horn extension problems, but addressing “outer horn" extensions requires a more elaborate framework.

At the top level, GAIA uses the simplicial category of ordinal numbers with objects defined as $[n], n \geqslant 0$ and arrows defined as weakly order-preserving mappings $f : [n] \to [m]$, where $f(i) \leqslant f(j), i \leqslant j$. This top-level structure can be viewed as a combinatorial “factory" for constructing, manipulating, and destructing complex objects that can be built out of modular components defined over categories. The second layer of GAIA defines the building blocks of generative AI models as universal coalgebras over categories that can be defined using current generative AI approaches, including Transformers that define a category of permutation-equivariant functions on vector spaces, structured state-space models that define a category over linear dynamical systems, or image diffusion models that define a probabilistic coalgebra over ordinary differential equations. The third layer in GAIA is a category of elements over a (relational) database that defines the data over which foundation models are built. GAIA formulates the machine learning problem of building foundation models as extending functors over categories, rather than interpolating functions on sets or spaces, which yields canonical solutions called left and right Kan extensions. GAIA uses the metric Yoneda Lemma to construct universal representers of objects in non-symmetric generalized metric spaces. GAIA uses a categorical integral calculus of (co)ends to define two families of generative AI systems. GAIA models based on coends correspond to topological generative AI systems, whereas GAIA systems based on ends correspond to probabilistic generative AI systems.

Much of this paper has been devoted to a theoretical study of the GAIA framework for generative AI. Of course, the actual implementation and testing of GAIA is ultimately the only proof of its practical utility as a computing framework for generative AI. We anticipate the problem of designing GAIA systems is a multi-year (possibly multi-decade!) project, since it requires harnessing many sophisticated mathematical ideas into practical algorithms and hardware implementations. What makes this challenge feasible is the existence of algorithms that solve very restricted types of machine learning problems that are already using similar technology such as used in GAIA. UMAP McInnes et al. [2018] is an elegant dimensionality reduction that constructs a simplicial set from high-dimensional image or textual data, and then constructs functors that map the data into a topological space. In effect, McInnes et al. [2018] construct a coend, although that is not the way the paper describes it. But as MacLane [1971] makes clear, topological realizations are in fact coend objects. The algorithm in UMAP works on data in $\mathbb{R}^n$ and it transparently extends to sequence data used by Transformer models, which lie in $\mathbb{R}^{d \times n}$. There are interesting questions in how to extend UMAP into a full GAIA model by using simplicial learning, which is a topic of a future paper.

References

P. Aczel. *Non-Well-Founded Sets*. CSLI Lecture Notes, Palo Alto, CA, USA, 1988.

S. A. Andersson, D. Madigan, and M. D. Perlman. A characterization of Markov equivalence classes for acyclic digraphs. *The Annals of Statistics*, 25(2):505 – 541, 1997. doi:10.1214/aos/1031833662. URL https://doi.org/10.1214/aos/1031833662.

T. Avery. Codensity and the giry monad. *Journal of Pure and Applied Algebra*, 220(3):1229–1251, Mar. 2016. ISSN 0022-4049. doi:10.1016/j.jpaa.2015.08.017. URL http://dx.doi.org/10.1016/j.jpaa.2015.08.017.

J. Baez and M. Stay. Physics, topology, logic and computation: A rosetta stone. In *New Structures for Physics*, pages 95–172. Springer Berlin Heidelberg, 2010. doi:10.1007/978-3-642-12821-9_2. URL https://doi.org/10.1007%2F978-3-642-12821-9_2.

J. Barwise and L. S. Moss. *Vicious circles - on the mathematics of non-wellfounded phenomena*, volume 60 of *CSLI lecture notes series*. CSLI, 1996. ISBN 978-1-57586-009-1.

Y. Bengio. Learning deep architectures for AI. *Foundations and Trends in Machine Learning*, 2(1):1–127, 2009.

D. Bertsekas. *Reinforcement Learning and Optimal Control*. Athena Scientific, 2019.

D. P. Bertsekas. *Dynamic programming and optimal control, 3rd Edition*. Athena Scientific, 2005. ISBN 1886529264. URL https://www.worldcat.org/oclc/314894080.

M. Boardman and R. Vogt. *Homotopy invariant algebraic structures on topological spaces*. Springer, Berlin, 1973.

R. Bommasani, D. A. Hudson, E. Adeli, R. Altman, S. Arora, S. von Arx, M. S. Bernstein, J. Bohg, A. Bosselut, E. Brunskill, E. Brynjolfsson, S. Buch, D. Card, R. Castellon, N. Chatterji, A. Chen, K. Creel, J. Q. Davis, D. Demszky, C. Donahue, M. Doumbouya, E. Durmus, S. Ermon, J. Etchemendy, K. Ethayarajh, L. Fei-Fei, C. Finn, T. Gale, L. Gillespie, K. Goel, N. Goodman, S. Grossman, N. Guha, T. Hashimoto, P. Henderson, J. Hewitt, D. E. Ho, J. Hong, K. Hsu, J. Huang, T. Icard, S. Jain, D. Jurafsky, P. Kalluri, S. Karamcheti, G. Keeling, F. Khani, O. Khattab, P. W. Koh, M. Krass, R. Krishna, R. Kuditipudi, A. Kumar, F. Ladhak, M. Lee, T. Lee, J. Leskovec, I. Levent, X. L. Li, X. Li, T. Ma, A. Malik, C. D. Manning, S. Mirchandani, E. Mitchell, Z. Munyikwa, S. Nair, A. Narayan, D. Narayanan, B. Newman, A. Nie, J. C. Niebles, H. Nilforoshan, J. Nyarko, G. Ogut, L. Orr, I. Papadimitriou, J. S. Park, C. Piech, E. Portelance, C. Potts, A. Raghunathan, R. Reich, H. Ren, F. Rong, Y. Roohani, C. Ruiz, J. Ryan, C. Ré, D. Sadigh, S. Sagawa, K. Santhanam, A. Shih, K. Srinivasan, A. Tamkin, R. Taori, A. W. Thomas, F. Tramèr, R. E. Wang, W. Wang, B. Wu, J. Wu, Y. Wu, S. M. Xie, M. Yasunaga, J. You, M. Zaharia, M. Zhang, T. Zhang, X. Zhang, Y. Zhang, L. Zheng, K. Zhou, and P. Liang. On the opportunities and risks of foundation models, 2022.

M. Bonsangue, F. van Breugel, and J. Rutten. Generalized metric spaces: Completion, topology, and powerdomains via the yoneda embedding. *Theoretical Computer Science*, 193(1):1–51, 1998. ISSN 0304-3975. doi:https://doi.org/10.1016/S0304-3975(97)00042-X. URL https://www.sciencedirect.com/science/article/pii/S030439759700042X.

F. Borceux. *Handbook of Categorical Algebra*, volume 1 of *Encyclopedia of Mathematics and its Applications*. Cambridge University Press, 1994. doi:10.1017/CBO9780511525858.

V. S. Borkar. *Stochastic Approximation: A Dynamical Systems Viewpoint*. Cambridge University Press, 2008.

T. Bradley, J. Terilla, and Y. Vlassopoulos. An enriched category theory of language: From syntax to semantics. *La Matematica*, 1:551–580, 2022.

G. Carlsson and F. Memoli. Classifying clustering schemes, 2010. URL http://arxiv.org/abs/1011.5270. cite arxiv:1011.5270.

G. J. Chaitin. *Exploring RANDOMNESS*. Discrete mathematics and theoretical computer science. Springer, 2002. ISBN 978-1-85233-417-8.

B. Coecke, T. Fritz, and R. W. Spekkens. A mathematical theory of resources. *Information and Computation*, 250: 59–86, oct 2016. doi:10.1016/j.ic.2016.02.008. URL https://doi.org/10.1016%2Fj.ic.2016.02.008.

T. M. Cover and J. A. Thomas. *Elements of Information Theory 2nd Edition (Wiley Series in Telecommunications and Signal Processing)*. Wiley-Interscience, July 2006. ISBN 0471241954.

R. Fagin, J. Y. Halpern, Y. Moses, and M. Y. Vardi. *Reasoning About Knowledge*. MIT Press, 1995. ISBN 9780262562003. doi:10.7551/MITPRESS/5803.001.0001. URL https://doi.org/10.7551/mitpress/5803.001.0001.

F. Feys, H. H. Hansen, and L. S. Moss. Long-Term Values in Markov Decision Processes, (Co)Algebraically. In C. Cîrstea, editor, *14th International Workshop on Coalgebraic Methods in Computer Science (CMCS)*, volume LNCS-11202 of *Coalgebraic Methods in Computer Science*, pages 78–99, Thessaloniki, Greece, Apr. 2018. Springer International Publishing. doi:10.1007/978-3-030-00389-0_6. URL https://inria.hal.science/hal-02044650.

B. Fong. Causal theories: A categorical perspective on bayesian networks, 2012.

B. Fong and D. I. Spivak. *Seven Sketches in Compositionality: An Invitation to Applied Category Theory*. Cambridge University Press, 2018.

B. Fong, D. I. Spivak, and R. Tuyéras. Backprop as functor: A compositional perspective on supervised learning. In *34th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2019, Vancouver, BC, Canada, June 24-27, 2019*, pages 1–13. IEEE, 2019. doi:10.1109/LICS.2019.8785665. URL https://doi.org/10.1109/LICS.2019.8785665.

T. Fritz and A. Klingler. The d-separation criterion in categorical probability. *Journal of Machine Learning Research*, 24(46):1–49, 2023. URL http://jmlr.org/papers/v24/22-0916.html.

P. Gabriel, P. Gabriel, and M. Zisman. *Calculus of Fractions and Homotopy Theory*. Calculus of Fractions and Homotopy Theory. Springer-Verlag, 1967. ISBN 9780387037776. URL https://books.google.com/books?id=UEQZAQAAIAAJ.

M. Gavrilovich. The unreasonable power of the lifting property in elementary mathematics, 2017. URL https://arxiv.org/abs/1707.06615.

A. Gu, K. Goel, and C. Ré. Efficiently modeling long sequences with structured state spaces. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022. URL https://openreview.net/forum?id=uYLFoz1vlAC.

A. Gu, I. Johnson, A. Timalsina, A. Rudra, and C. Ré. How to train your HIPPO: state space models with generalized orthogonal basis projections. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023. URL https://openreview.net/pdf?id=klK17OQ3KB.

J. Y. Halpern. *Actual Causality*. MIT Press, 2016. ISBN 978-0-262-03502-6.

B. Jacobs. *Introduction to Coalgebra: Towards Mathematics of States and Observation*, volume 59 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, 2016. ISBN 9781316823187. doi:10.1017/CBO9781316823187. URL https://doi.org/10.1017/CBO9781316823187.

B. Jacobs, A. Kissinger, and F. Zanasi. Causal inference by string diagram surgery, 2019.

A. Joyal. Quasi-categories and kan complexes. *Journal of Pure and Applied Algebra*, 175(1):207–222, 2002. ISSN 0022-4049. doi:https://doi.org/10.1016/S0022-4049(02)00135-4. URL https://www.sciencedirect.com/science/article/pii/S0022404902001354. Special Volume celebrating the 70th birthday of Professor Max Kelly.

D. Kan. Adjoint functors. *Transactions of the American Mathematical Society*, 87(2):294–329, 1958. URL https://doi.org/10.2307/1993102.

D. Kozen and N. Ruozzi. Applications of metric coinduction. *Log. Methods Comput. Sci.*, 5(3), 2009. URL http://arxiv.org/abs/0908.2793.

H. Kushner and G. Yin. *Stochastic Approximation and Recursive Algorithms and Applications*. Stochastic Modelling and Applied Probability. Springer New York, 2003. ISBN 9780387008943. URL https://books.google.com/books?id=_0bIieuUJGkC.

J. Liu, J. Chen, and J. Ye. Large-scale sparse logistic regression. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 547–556. ACM, 2009.

F. Loregian. *(Co)end Calculus*. London Mathematical Society Lecture Note Series. Cambridge University Press, 2021. doi:10.1017/9781108778657.

J. Lurie. *Higher Topos Theory*. Annals of mathematics studies. Princeton University Press, Princeton, NJ, 2009. URL https://cds.cern.ch/record/1315170.

J. Lurie. Kerodon. https://kerodon.net, 2022.

S. MacLane. *Categories for the Working Mathematician*. Springer-Verlag, New York, 1971. Graduate Texts in Mathematics, Vol. 5.

S. MacLane and leke Moerdijk. *Sheaves in Geometry and Logic: A First Introduction to Topos Theory*. Springer, 1994.

S. Mahadevan. Causal homotopy, 2021a. URL https://arxiv.org/abs/2112.01847.

S. Mahadevan. Universal decision models. *CoRR*, abs/2110.15431, 2021b. URL https://arxiv.org/abs/2110.15431.

S. Mahadevan. Categoroids: Universal conditional independence, 2022. URL https://arxiv.org/abs/2208.11077.

S. Mahadevan. Universal causality. *Entropy*, 25(4):574, 2023. doi:10.3390/E25040574. URL https://doi.org/10.3390/e25040574.

J. May. *Simplicial Objects in Algebraic Topology*. University of Chicago Press, 1992.

J. May and K. Ponto. *More Concise Algebraic Topology: Localization, Completion, and Model Categories*. Chicago Lectures in Mathematics. University of Chicago Press, 2012. ISBN 9780226511788. URL https://books.google.com/books?id=SHhmxUPskFwC.

L. McInnes, J. Healy, and J. Melville. Umap: Uniform manifold approximation and projection for dimension reduction, 2018. URL https://arxiv.org/abs/1802.03426.

M. Papillon, S. Sanborn, M. Hajij, and N. Miolane. Architectures of topological deep learning: A survey on topological neural networks, 2023.

J. Pearl. *Causality: Models, Reasoning and Inference*. Cambridge University Press, USA, 2nd edition, 2009. ISBN 052189560X.

D. G. Quillen. *Homotopical algebra*. Springer, 1967.

B. Richter. *From Categories to Homotopy Theory*. Cambridge Studies in Advanced Mathematics. Cambridge University Press, 2020. ISBN 9781108479622. URL https://books.google.com/books?id=pnzUDwAAQBAJ.

E. Riehl. *Category Theory in Context*. Aurora: Dover Modern Math Originals. Dover Publications, 2017. ISBN 9780486820804. URL https://books.google.com/books?id=6B9MDgAAQBAJ.

H. Robbins and S. Monro. A Stochastic Approximation Method. *The Annals of Mathematical Statistics*, 22(3):400 – 407, 1951. doi:10.1214/aoms/1177729586. URL https://doi.org/10.1214/aoms/1177729586.

J. Rutten. Universal coalgebra: a theory of systems. *Theoretical Computer Science*, 249(1):3 – 80, 2000. ISSN 0304-3975. doi:http://dx.doi.org/10.1016/S0304-3975(00)00056-6. URL http://www.sciencedirect.com/science/article/pii/S0304397500000566. Modern Algebra.

B. Schölkopf and A. J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, 2002.

A. Sokolova. Probabilistic systems coalgebraically: A survey. *Theoretical Computer Science*, 412(38):5095–5110, 2011. ISSN 0304-3975. doi:https://doi.org/10.1016/j.tcs.2011.05.008. URL https://www.sciencedirect.com/science/article/pii/S0304397511003902. CMCS Tenth Anniversary Meeting.

Y. Song and S. Ermon. Generative modeling by estimating gradients of the data distribution. In H. M. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. B. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 11895–11907, 2019. URL https://proceedings.neurips.cc/paper/2019/hash/3001ef257407d5a371a96dcd947c7d93-Abstract.html.

D. I. Spivak. Database queries and constraints via lifting problems. *Mathematical Structures in Computer Science*, 24 (6), oct 2013. doi:10.1017/s0960129513000479. URL https://doi.org/10.1017%2Fs0960129513000479.

D. I. Spivak and R. E. Kent. Ologs: A categorical framework for knowledge representation. *PLoS ONE*, 7(1):e24274, jan 2012. doi:10.1371/journal.pone.0024274.

M. Studeny. *Probabilistic Conditional Independence Structures*. Information Science and Statistics. Springer London, 2010. ISBN 9781849969482. URL https://books.google.com.gi/books?id=bGFRcgAACAAJ.

R. S. Sutton and A. G. Barto. *Reinforcement learning - an introduction*. Adaptive computation and machine learning. MIT Press, 1998. ISBN 978-0-262-19398-6. URL https://www.worldcat.org/oclc/37293240.

A. Turing. Computing machinery and intelligence. *Mind*, 49:433–460, 1950.

A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. In I. Guyon, U. von Luxburg, S. Bengio, H. M. Wallach, R. Fergus, S. V. N. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5998–6008, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html.

S. Vigna. A guided tour in the topos of graphs, 2003.

C. Villani. *Topics in Optimal Transportation*. American Mathematical Society, 2003.

R. Vollmar. John von neumann and self-reproducing cellular automata. *J. Cell. Autom.*, 1(4):353–376, 2006. URL http://www.oldcitypublishing.com/journals/jca-home/jca-issue-contents/jca-volume-1-number-4-2006/jca-1-4-p-353-376/.

E. Wagstaff, F. B. Fuchs, M. Engelcke, M. A. Osborne, and I. Posner. Universal approximation of functions on sets. *J. Mach. Learn. Res.*, 23:151:1–151:56, 2022. URL http://jmlr.org/papers/v23/21-0730.html.

S. Wolfram. *A new kind of science*. Wolfram-Media, 2002. ISBN 978-1-57955-008-0.

D. Yarotsky. Universal approximations of invariant maps by neural networks. *CoRR*, abs/1804.10306, 2018. URL http://arxiv.org/abs/1804.10306.

T. Yin, M. Gharbi, R. Zhang, E. Shechtman, F. Durand, W. T. Freeman, and T. Park. One-step diffusion with distribution matching distillation, 2023.

N. Yoneda. On ext and exact sequences. *J. Fac. Sci. Univ. Tokyo*, Sect. I 8:507–576, 1960.

C. Yun, S. Bhojanapalli, A. S. Rawat, S. J. Reddi, and S. Kumar. Are transformers universal approximators of sequence-to-sequence functions? In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. URL https://openreview.net/forum?id=ByxRM0Ntvr.