SOME MATHEMATICIANS ARE BIRDS, OTHERS ARE FROGS. BIRDS FLY HIGH IN THE AIR AND SURVEY BROAD VISTAS OF MATHEMATICS OUT TO THE FAR HORIZON. THEY DELIGHT IN CONCEPTS THAT UNIFY OUR THINKING AND BRING TOGETHER DIVERSE PROBLEMS FROM DIFFERENT PARTS OF THE LANDSCAPE. FROGS LIVE IN THE MUD BELOW AND SEE ONLY THE FLOWERS THAT GROW NEARBY. THEY DELIGHT IN THE DETAILS OF PARTICULAR OBJECTS, AND THEY SOLVE PROBLEMS ONE AT A TIME.

FREEMAN DYSON, NOTICES OF THE AMS, VOL. 56, NO 2.

SRIDHAR MAHADEVAN

CATEGORIES FOR AGI

PUBLISHER OF THIS BOOK

First printing, February 2026

Contents

List of Figures

List of Tables

I propose to consider the question, "Can machines think?".

A. M. Turing (1950) , Mind 49: 433-460.

Introduction

This book is intended to serve as lecture notes for `CMPSCI 692CT: Category Theory for AGI`, taught by the author during Spring 2026 at the University of Massachusetts, Amherst. It is intended to bridge the gap between formal books on category theory, written by mathematicians for mathematicians [1], and the need for a more approachable book for AI researchers wanting to apply these abstract ideas to Artificial General Intelligence (AGI).

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

The world's largest AI companies are collectively spending several trillion dollars in the most expensive race in human history to build AGI.[2] This course will give students a detailed introduction to category theory, and how to use it to analyze today's AGI systems, understand their limitations and how to design the next generation of AGI systems. We'll cover the core theoretical concepts, including categories, functors, natural transformations, the Yoneda lemma, limits and colimits, adjunctions, monads, and Kan extensions, as well as their application to building AGI systems that can reason causally, learn from their experience, plan to achieve long-term goals, interact with users in natural language, and ultimately, achieve consciousness.

[2] "We could hit a wall: why trillions of dollars of risk is no guarantee of AI reward", Dan Milmo, The Guardian, January 17, 2026, `http://bit.ly/4qROBTg`

The book assumes no prior knowledge of category theory, but does require the reader to have sufficient knowledge of basic mathematics – including multivariate calculus, probability and statistics, modern algebra, and graph theory – as well as sufficient background in artificial intelligence (AI), machine learning (ML), and a working knowledge of computer programming, particularly `Python`.

The book is not intended to serve as a comprehensive overview of AI, ML, or category theory, but will reflect the author's own interests and research activity.[3] The reader should be familiar in using modern AI coding tools, such as `chatGPT` or `Claude Code`, and have access to computing facilities that will be required in running the sample code provided, and do the required final project (e.g., `Google Collab` is free for all students).

[3] Throughout this book, you will find suggested exercises in each chapter. These are intended to test your understanding of the basic concepts.

Category Theory for AGI

AGI is possibly the grandest project ever conceived by humanity: create a computer program that can automate human cognitive abilities across the complete spectrum, from perception to language, and be capable of acquiring world knowledge at a massive scale from consuming all of humanity's digital (and analog) footprint over millennia.[4] It is a remarkable tribute to the success of AI and ML, as well as the incredible power of modern computing devices, that this goal seems achievable, not in the distant future, but in the near-term.

[4] https://cloud.google.com/discover/what-is-artificial-general-intelligence

ALAN TURING[5] formalized the problem of AGI in terms of an *imitation game*:

[5] Alan Turing. Computing machinery and intelligence. *Mind*, 49:433–460, 1950

> The new form of the problem can be described in terms of a game which we call the 'imitation game." It is played with three people, a man (A), a woman (B), and an interrogator (C) who may be of either sex. The interrogator stays in a room apart front the other two. The object of the game for the interrogator is to determine which of the other two is the man and which is the woman. He knows them by labels X and Y, and at the end of the game he says either "X is A and Y is B" or "X is B and Y is A."
>
> In order that tones of voice may not help the interrogator the answers should be written, or better still, typewritten. The ideal arrangement is to have a teleprinter communicating between the two rooms. Alternatively the question and answers can be repeated by an intermediary. The object of the game for the third player (B) is to help the interrogator. The best strategy for her is probably to give truthful answers. She can add such things as "I am the woman, don't listen to him!" to her answers, but it will avail nothing as the man can make similar remarks. We now ask the question, "What will happen when a machine takes the part of A in this game?" Will the interrogator decide wrongly as often when the game is played like this as he does when the game is played between a man and a woman? These questions replace our original, "Can machines think?"

We will use Turing's definition of AGI as our working hypothesis: a system can be viewed as achieving AGI if it cannot be accurately discriminated from humans through remote interaction.

AGI as Consciousness

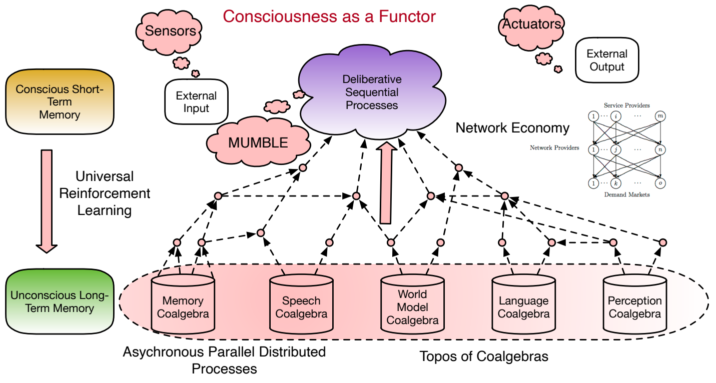


Figure 1: An AGI Architecture for Consciousness

We set ourself an ambitious goal in this course and book to ultimately attempt to model consciousness mathematically using the tools of category theory. Humans are conscious: what this means is that at any given moment of time, we are aware of a very small number of things that are placed in our *short-term* conscious memory, and unaware of the maelstrom of unconscious processes that are asynchronously running in our long-term memory. We can view AGI as an attempt to not just capture the input-output behavior of humans and other animals, but also to understand the origins of consciousness. Will sufficiently advanced AGI systems attain consciousness?

CONSCIOUSNESS balances slow deliberative short-term memory with fast asynchronous long-term memory. When you learn a new skill – such as playing a piano – your short-term conscious memory is hard at work. Each note requires careful attention as to where to place your fingers. A few years of practice, and what once took immense concentration becomes now an automatic behavior, compiled into long-term memory. The behavior is now asynchronous, parallel, distributed, fluid and you are not aware of the specific movements of your fingers anymore.

The above figure shows a model of consciousness from a recent paper of mine.[6] It builds on a lot of sophisticated ideas that we will cover in this course, including *topos theory* – a set-like category that enables internal languages of "thought" – and modeling dynamical systems as *universal coalgebras*, and generalized multi-agent decision making as a *network economy*. These components will be progressively studied through the semester as we learn the mathematical tools that are described in this model of consciousness.

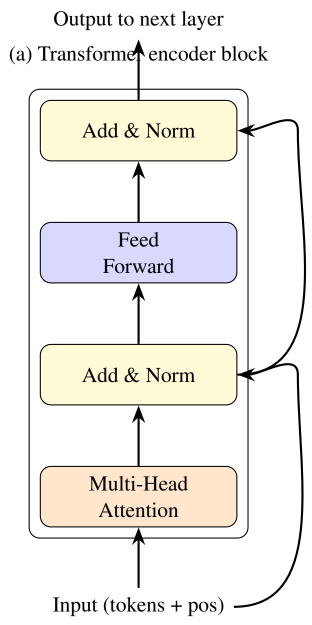
[6] Sridhar Mahadevan. Consciousness as a functor, 2025a. URL https://arxiv.org/abs/2508.17561

We will return to the question of consciousness at the end of this book, but we will keep this broad goal in mind through the course and book. To begin more concretely, however, let us start with what is currently achievable using modern AI and ML technology.

Transformers

GPT (Generalized Pretrained Transformer) models have revolutionized AI in the last few years, and arguably are the closest any computer program has come to achieving AGI. In this book and course, we will study GPT models through the mathematical framework of category theory. The original Transformer model [7] – now possibly the most cited paper in AI – can be abstractly visualized by the following diagram.

[7] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman 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



TRANSFORMER models are the foundation of modern AGI: we will spend considerable effort in understanding this architecture during this course. If we are to build a better Transformer, how we go about it? There are over 1 *million* implementations of Transformer models available online [8]. So, plenty of programmers and researchers have already explored this model, undoubtedly due to its enormous influence on today's AI systems. What makes our book and course different? There are many ways to understand Transformer models. One obvious way, which you already have probably mastered, is at the architectural level by viewing it as a *deep learning* model [9] or as an *natural language processing* (NLP) system [10].

[8] https://huggingface.co/models?library=transformers&sort=trending.

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

[10] Tong Xiao and Jingbo Zhu. Introduction to Transformers: an NLP perspective, 2023. URL https://arxiv.org/abs/2311.17633

To motivate the study of Transformers, consider the architectural diagram shown above, which defines the classic `PostLN` model: the *Layer Normalization* (LN) module is placed after the `Multi-Head Attention` (MHA) module. Experience over the past few years has shown that this particular architecture is difficult to train, requiring careful adjustment of *learning rates*. In contrast, the `PreLN` Transformer architecture places the LN module before the MHA module, making it significantly easier to train.[11] To understand the possible space of different architectures and their impact, we will bring in a more theoretical framework in this course.

[11] Jeonghoon Kim, Byeongchan Lee, Cheonbok Park, Yeontaek Oh, Beomjun Kim, Taehwan Yoo, Seongjin Shin, Dongyoon Han, Jinwoo Shin, and Kang Min Yoo. Peri-ln: Revisiting normalization layer in the transformer architecture, 2025. URL https://arxiv.org/abs/2502.02732

Transformers and Topos Categories

TRANSFORMER models will be studied in this course building on the abstract concepts in category theory, in particular a *topos* category. [12]

[12] Review the basic definition of categories from Riehl's book Section 1.1

Definition 1. *A category $\mathcal{C}$ is a collection of abstract* objects $c \in \mathcal{C}$. *Anything technically can count as an object, from a layer in a Transformer model to an entire model itself. Each category $\mathcal{C}$ is additionally specified by a set of arrows or morphisms $\mathcal{C}(c, d)$ between each pair of objects c and d. There is an identity arrow $1_c \in \mathcal{C}(c, c)$. Arrows compose in the obvious way, inducing a function $\mathcal{C}(c, d) \times \mathcal{C}(d, e) \rightarrow \mathcal{C}(c, e)$.*

When you read this definition, you should try to construct a few examples of categories. An *initial object* c in category $\mathcal{C}$ is defined as one inducing a unique arrow from c to every object in category $\mathcal{C}$. A *terminal* object, usually denoted by $\mathbf{1}$, is one that defines a unique arrow from every object c in category $\mathcal{C}$ into $\mathbf{1}$. Not all categories have initial or final objects. [13] An object c is isomorphic to another object d, denoted $c \simeq d$, if two arrows $f : c \rightarrow d$ and $g : d \rightarrow c$ exist, such that $g \circ f = 1_c$, and $f \circ g = 1_d$.

[13] Exercise: Consider the category of **Sets**. Does it have a terminal object? An initial object? What are they, if they exist?

A word on notation. In many category theory books or papers, you will see two choices for referring to the set of arrows $\mathcal{C}(a, b)$, or $\mathbf{Hom}_{\mathcal{C}}(a, b)$. Both of these denote the set of arrows from objects a to b in category $\mathcal{C}$. In most applications to AGI, we actually impose some additional structure on this set, which makes it an *enriched* category. That is, $\mathcal{C}(a, b)$ is not merely a set, but it can be a vector space (e.g., each arrow is a linear transformation between vector spaces). For Transformer models, for example, if a and b denote Transformer models, or layers of Transformer models, then we can see that the arrows between two Transformer models must have additional structure as well.

WHERE TRANSFORMERS ENTER. A trained Transformer is a concrete realizable approximation to a sequence-to-sequence function $f_\theta : \mathbb{R}^{n \times d} \rightarrow \mathbb{R}^{n \times d}$. Thus, we can regard Transformer models as inhabiting (or approximating) arrows in a function-based topos-like universe. This viewpoint is not a claim that "the set of all neural networks" is itself an elementary topos without further structure; rather it is a guiding semantics that will become operational when we introduce patch sites, sheaves, and internal languages later in the course.

TOPOS theory can give us a powerful guide to designing a novel class of

Transformer models [14]. To give some intuition, let us consider a simple example of a topos category, where every object is simply a function $f : A \to B$, from some arbitrary domain set A to some co-domain set B. To define a category of functions on sets, we need to specify both the objects of the category – this is straightforward, as it is simply every possible function – as well as the collection of arrows between every pair of objects. This appears less straightforward: how do we define arrows between function objects?

[14] Sridhar Mahadevan. Topos theory for generative AI and LLMs, 2025e. URL https://arxiv.org/abs/2508.08293

COMMUTATIVE DIAGRAMS are an essential data structure for understanding categories. They are defined like graphs, with vertices representing objects, and edges representing arrows. To define arrows between our topos category of Transformer function objects, we use a commutative diagram such as the one illustrated below:

$$\begin{tikzcd} A \arrow[r, "h"'] \arrow[d, "f"] & A' \arrow[d, "f'"'] \\ B \arrow[r, "g"] & B' \end{tikzcd}$$

Here, A, A', B and B' denote sets, and f, f', g and h define functions between a pair of sets as shown in the diagram. This diagram should be viewed as defining a set of constraints. It asserts that the *composition* of functions $g \circ f$ (meaning, first apply f and then apply g) is equivalent to the composition $f' \circ h$. So, how does this give us a collection of arrows in our topos function category?

A PRECISE TOY TOPOS. There is a mathematically clean way to package "functions as objects" into an elementary topos: if $\mathcal{E}$ is an elementary topos (e.g., **Set**), then its *arrow category* $\mathcal{E}^{[1]}$—whose objects are arrows $f : A \to B$ and whose morphisms are commutative squares—is again an elementary topos. We will use $\mathbf{Set}^{[1]}$ as our running toy model.

YONEDA EMBEDDINGS: In chapter 3, we will encounter the beautiful Yoneda Lemma, which is the easiest way to build a topos from scratch. For any given category $\mathcal{C}$, map any object c in it to the set-valued functor $\mathcal{C}(-, c)$ (note that plugging in any object d into the $-$ gives us a set!). This embedding has the most remarkable properties, and it takes a whole book to explain them. [15]

[15] Saunders Mac Lane and Ieke Moerdijk. *Sheaves in Geometry and Logic a First Introduction to Topos Theory*. Springer New York, New York, NY, 1992. ISBN 9781461209270 1461209277. URL http://link.springer.com/book/10.1007/978-1-4612-0927-0

MORPHISMS BETWEEN FUNCTION-OBJECTS. In the arrow category $\mathbf{Set}^{[1]}$, a morphism from $f : A \to B$ to $f' : A' \to B'$ is precisely a pair of functions $h : A \to A'$ and $g : B \to B'$ such that $g \circ f = f' \circ h$. That is, morphisms are commutative squares.

We view the objects of our topos category all functions $f : A \to B$, and for each pair of such objects, such as f and f', we define an arrow to mean the pair $\langle g, h\rangle$ such that constraints imposed by the above commutative diagram are satisfied. Notice that between any particular pair of objects f and f', there may be many – in fact, possibly an infinite number – of arrows. It is also obvious that the number of objects need not be finite. But, just as we are comfortable manipulating real numbers on computers, which in fact are the basis for deep learning and Transformer models, despite the fact that the set of real numbers is not even countable, we should be comfortable in dealing with categories with an infinite number of objects and arrows.

TOPOSES are defined by a set of conditions, which can be found in any standard textbook on topos theory. First, let us understand how this abstract categorical structure applies to Transformer models. Recall that each Transformer model can be viewed as a mapping from a sequence of tokens in $\mathbb{R}^{n\times d}$ to another sequence of tokens in $\mathbb{R}^{n\times d}$. A well-known result shows that Transformer models are *universal function approximators* over such sequences. [16] Intuitively, this result implies that for any function over $\mathbb{R}^{n\times d}$, there exists a Transformer model that is within some small ϵ distance in a normed vector space. So, we can squint our eyes a bit, and pretend that we can construct a Transformer model for any desired function in this class.

[16] Chulhee Yun, Srinadh Bhojanapalli, Ankit Singh Rawat, Sashank J. Reddi, and Sanjiv 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

To show that our category of function objects with its arrow structure defined by these square diagrams forms a topos, we need to check whether all the conditions of a topos are satisfied. These conditions require introducing some fairly esoteric definitions, but to keep it simple, we focus on the application of these ideas to Transformer models. A central property of a topos is the ability to form pieces of an object, which are formally called subobjects. So, if each object is a Transformer model, we need to figure out how to construct subobjects from Transformer objects. Think of subobjects akin to subsets of a set. We can always take a set, and find a subset of it by selecting a few elements of the set. Notice that we can define a subset $X \subseteq Y$ in an alternative way, through its induced *characteristic function* $\psi_X : Y \to \{0,1\}$. An element $x \in Y$ is mapped to $\psi_X(y) = 1$ if it is a member of X, otherwise it is mapped to 0. We can generalize this concept to define subobjects of arbitrary objects in a category.

To help build some intuition, consider how to define subsets without "looking inside" a set. Essentially, a subset S of some larger set T can be viewed as a "monic arrow" (an injective function). Monic arrows generalize injective functions. An arrow $f : a \to b$ in a category $\mathcal{C}$ is called **monic** if given any parallel arrows $g, h : c \rightrightarrows a$, the equality $f \circ g = f \circ h$ implies that $g = h$, namely f is "left-cancellable". Now,

Definition 2. *In a category $\mathcal{C}$, a* **subobject classifier** *is a $\mathcal{C}$-object Ω, and a $\mathcal{C}$-arrow* $\mathbf{true} : \mathbf{1} \to \Omega$*, such that to every monic arrow $S \hookrightarrow X$ in $\mathcal{C}$, there is a unique arrow ϕ that forms the following pullback square commutative* *diagram:*

$$\begin{tikzcd} S \arrow[r] \arrow[d, tail, "m"] & \mathbf{1} \arrow[d, tail, "\textbf{true}"] \\ X \arrow[r, dashed, "\phi"] & \Omega \end{tikzcd}$$

What does this diagram mean? Consider the category of **Sets**. Let S be some subset of X. Then, m defines a monic (1-1) function that simply maps each element $x \in S$ to the same element $x \in X$. ϕ is the characteristic (Boolean) function that maps X to a Boolean $\{0, 1\}$-valued object Ω. The arrow **true** simply maps **1** to the element 1 in Ω. The diagram commutes, hence for each element $x \in X$, it must be the case that composing the **true** with the unique function mapping S to **1** (which recall is a terminal object) gives us the exact same arrow as composing ϕ with m.

This commutative diagram enforces the constraint that every monic arrow m (i.e., every $1 - 1$ function) that maps a subobject S to an object X must be characterizable in terms of a “pullback", a particular type of universal property that is a special type of a limit. In the special case of the category of sets, subobject classifiers are defined through the characteristic (Boolean-valued) function ϕ that defines subsets. In general, such as for Transformer objects, the subobject classifier Ω is not Boolean-valued, and requires using intuitionistic logic through a Heyting algebra.

Remarkably, we can generalize this definition to our Transformer category as well, if we view Transformers as general sequence-to-sequence functions. We will explain the details of this construction later in this course, but for those impatient readers, the proof is given here. [17] The proof is actually quite simple: it requires constructing a commutative diagram similar to that shown above, and we urge the reader to consider proving this result before looking at this paper.

[17] Sridhar Mahadevan. Topos theory for generative AI and LLMs, 2025e. URL https://arxiv.org/abs/2508.08293

Definition 3. *A category $\mathcal{C}$ has* **binary products** *if for every pair of objects, c and d, there exists a third object, $e \simeq c \times d$, along with two projection arrows, $p_1 : e \to c$ and $p_2 : e \to d$, such that for any other object a and arrows $f : a \to c$ and $g : a \to d$, there exists a unique morphism $u : a \to e$ satisfying $p_1 \circ u = f$ and $p_2 \circ u = g$.*

This abstract definition is a lot like many others you will encounter in this book. So, it takes some patience to work through what it means. [18] So, intuitively, given two Transformer objects, we need to be able to construct a “product" Transformer that satisfies the above definition. If we think of a Transformer as a sequence-to-sequence function, then we simplify the problem to thinking about how to define the product of two sequence-to-sequence functions. In addition to product objects, a topos category must have *exponential* objects.

[18] Exercise: draw the commutative diagram for this definition.

Definition 4. *A category $\mathcal{C}$ with binary products has* **exponential objects**

if for each pair of objects c, d in $\mathcal{C}$, there exists an object c^d that defines the following bijection:

$$\mathcal{C}(e \times d, c) \simeq \mathcal{C}(e, c^d)$$

Consider, for example, the category of **Sets**. Does it have exponential objects? [19] For our category of Transformer objects, how would we construct exponential objects? We will explore that question later in this book, but for now, let us assume this is possible. We can then proceed to define yet another property of a topos category.

[19] Hint: for each pair of sets X and Y, consider all functions $f : X \to Y$. Does this define an exponential object?

EXAMPLE (SETS). In **Set**, the exponential object Y^X can be taken to be the set of all functions $X \to Y$. There is a canonical *evaluation* map

$$\text{ev} : Y^X \times X \to Y, \qquad \text{ev}(f, x) = f(x),$$

and the universal property of Y^X states that every function $F : E \times X \to Y$ corresponds uniquely to its curried form $\widetilde{F} : E \to Y^X$. We will repeatedly reuse this pattern when we interpret modules inside Transformer architectures as "function objects" and later when we interpret patchwise reasoning as internal higher-order structure.

Definition 5. *A category $\mathcal{C}$ is* **Cartesian closed** *if it has binary products, a terminal object* **1**, *and exponential objects.*

With these definitions behind us, we can finally now state the key property of a topos category. As we will see, Transformers indeed do form a topos category, and with that deeper understanding, we can unlock a powerful novel class of Transformer architectures.

Definition 6. *An* **elementary topos** *is a category $\mathcal{C}$ that is Cartesian closed and has a subobject classifier.*

For example, the category of sets forms a topos. Binary products exist because one can define Cartesian products of sets. Exponential objects correspond to the set of all functions between two sets. The terminal object is simply the single-element set $\{\bullet\}$. Finally, the subobject classifier is simply the subset function, which induces a boolean-valued characteristic function.

THE TABLE below summarizes the differences between set theory and topos theory. We have seen some of these properties in this chapter, but we will our exploration of other properties in the subsequent chapters, beginning with the crucial concept of *functors* in the next chapter.

Set theory	**Topos 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

Summary and Further Reading

In this introductory chapter, we introduced the problem of AGI, and described how the Transformer model – the centerpiece of the current AI revolution – can be understood as a type of topos category. For many of you encountering these ideas for the first time, it can take quite a bit of time to process the meaning of these definitions. It is highly recommended that you read Section 1.1-1.2 of Riehl's textbook on a more formal definition of categories [20]. She goes through many examples of concrete categories. A canonical example is that of vector spaces, which plays a central role in Transformer models, as the space of input and output tokens $\mathbb{R}^{n \times d}$ forms a category with many special properties.

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

TRANSFORMER model implementations are widely available on the web. A great source is the implementation that comes with the O'Reilly book `Natural Language Processing with Transformers`.[21] I highly recommend running the sample notebooks in this GitHub repo, particularly the basic ones that walk you through how the Transformer model works. This course assumes you have sufficient knowledge through reading and experimenting with basic Transformer models to follow the more advanced ideas that we will be exploring in the remaining chapters.

[21] https://github.com/nlp-with-transformers

Functors for AGI

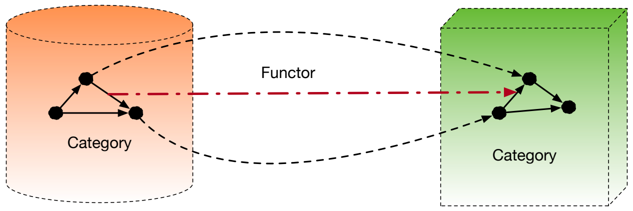


Figure 2: Functors act on both the objects and arrows of a category.

FUNCTORS are fundamental to categorical AGI, and we devote this chapter to a deeper study of this mapping between categories. Section 1.3 in Riehl's textbook gives a great introduction to functors. [22] Functors act in a way that is different from a function. It is central to grasping the core ideas of category theory, and the reader is encouraged to spend time understanding this fundamental concept.

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

Definition 7. *A* functor $F : \mathcal{C} \to \mathcal{D}$ *between two categories* $\mathcal{C}$ *and* $\mathcal{D}$ *is specified by an* object function *mapping each* $c \in \mathcal{C}$ *to* $Fc \in \mathcal{D}$*, and an* arrows function *mapping each arrow* $f \in \mathcal{C}(c, d)$ *to* $Ff \in \mathcal{D}(Fc, Fd)$.

To appreciate this definition, let us pause for a moment and understand what it is saying. A functor is not a function: it acts both on the objects of the domain category, as well as its arrows. That seems like a small change in the notion of a function, but it is such apparently minor variations that can completely transform how we think of AI and ML.[23] We will go through a variety of examples of functors in this chapter and the remainder of this book. The simplest kind of functor, and in many ways, perhaps one of the most powerful, is the `Powerset` functor $\mathbb{P} : X \to 2^X$. It acts on the category **Sets**, and is an example of an *endofunctor*, as the output co-domain category is the same as the input domain category.

[23] Sridhar Mahadevan. Rethinking AI: From functions to functors. In *Proceedings of the Fortieth AAAI Conference on Artificial Intelligence, January 20-27, 2026, Singapore*, 2026. URL https://people.cs.umass.edu/~mahadeva/papers/AAAI_2026_SM_Talk_Rethinking_AI.pdf

Example 1. $\mathbb{P}(\{0,1\}) = \{\emptyset, \{0\}, \{1\}, \{0,1\}\}$.

To appreciate why $\mathbb{P}$ is indeed a functor, we must specify not only how it acts on objects, namely sets, as above, but also how it acts on arrows, namely functions on sets. Given any function $f : A \to B$, it follows straightforwardly that $\mathbb{P}(f) = \mathbb{P}(A) \to \mathbb{P}(B)$.[24]

[24] The reader is encouraged to work out simple examples of the Powerset functor.

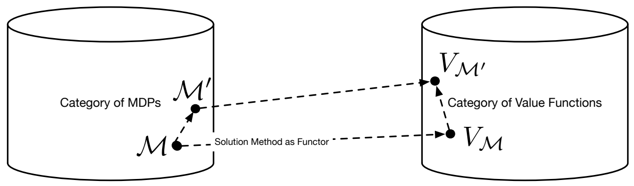


Figure 3: Reinforcement Learning (RL) methods can be modeled as functors.

Reinforcement Learning (RL) algorithms can be viewed as functors that map from the category $\mathcal{C}_{\text{MDP}}$ of Markov decision processes (MDPs) into the category $\mathcal{C}_V$ of value functions $V : S \to \mathbb{R}$. To understand why an RL method is a functor, consider computing the value function associated with some policy $\pi : S \to A$, which results in the value function V^π. Now, the arrows of category $\mathcal{C}_{\text{MDP}}$ are MDP *homomorphisms* – ways of abstracting an MDP that collapse its state space or action space. So, for any MDP homomorphism $\phi : M \to M'$, we get a corresponding mapping $V^\pi_M \to V^\pi_{M'}$. Functors act differently from functions: they map both the objects and arrows of a domain category into a co-domain category.

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.

Definition 8. *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 9. *A* **contravariant functor** *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, $F(1_c) = 1_{Fc}$.

We will use the notion $\mathcal{C}^{op}$ to denote the category $\mathcal{C}$ with the same objects, but with arrows reversed.

FUNCTOR CATEGORIES will play a major role in this book, and provide one of the most powerful types of categories, as they are central to the famous *Yoneda Lemma*. The functor category of *presheaves* and *copresheaves* are central to modeling Transformers.

Example 2. *The* functor category *of* presheaves $\mathbf{Sets}^{\mathcal{C}^{op}} : \mathcal{C}^{op} \rightarrow \mathbf{Sets}$ *is defined as a category, where every object is a presheaf, namely a mapping from any object* $c \in \mathcal{C}$ *to* $\mathbf{Sets}$.

Example 3. *The* Yoneda embedding *is defined as the mapping* $よ : \mathcal{C} \rightarrow \mathbf{Sets}^{\mathcal{C}^{op}}$*, and is simply defined as* $\mathcal{C}(-, c)$ *for any object* $c \in \mathcal{C}$.

Pay close attention to this powerful, and perhaps most beautiful of all embeddings in pure mathematics. What is it saying? Simply put, it maps an object, e.g., a word or token in a Transformer model, to the set of all sequences of tokens that precede it. In fact, that is precisely what the Transformer model learns from data. [25]

[25] Tai-Danae Bradley, John Terilla, and Yiannis Vlassopoulos. An enriched category theory of language: from syntax to semantics, 2021. Arxiv

Example 4. *The Yoneda embedding* $よ : \mathcal{C} \rightarrow \mathbf{Sets}^{\mathcal{C}^{op}}$ *creates a contravariant functor, because for any object* c*, its Yoneda embedding* $\mathcal{C}(-, c)$ *is a contravariant functor.*

This example is highly instructive: spend time on understanding why Yoneda embeddings create contravariant functors. [26]

[26] Hint: take any arrow $f : c \rightarrow c'$ in a category $\mathcal{C}$. How does the Yoneda embedding act on such arrows?

NATURAL TRANSFORMATIONS in fact turned out to be the fundamental reason why functors were defined, which in turn required inventing the concept of a category.

Definition 10. *Given any two functors* $F : C \rightarrow D$ *and* $G : C \rightarrow D$ *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.*

$$\begin{tikzcd}
Fc \arrow[r, "\alpha_c"] \arrow[d, "Ff"'] & Gc \arrow[d, "Gf"] \\
Fc' \arrow[r, "\alpha_{c'}"'] & Gc'
\end{tikzcd}$$

NATURAL TRANSFORMATIONS AND FUNCTORS are two of the most important aspects of categorical AGI, and it takes some effort to understand their importance. Rather than delve completely in the technicalities of their definition, let us understand first their crucial role in transforming AI. Fundamentally, modern AI is based around *functions*. We think of Transformer models as functions mapping sequences of tokens into other sequences of tokens, e.g., translating sentences in English into French or Japanese. Why do we need functors or natural transformations? The goal of this book and course is to explain their importance to AI and ML. At the end of this course, you will hopefully understand the importance of functors, and wonder why more attention is not being paid to them in AI research.

To begin with, let us ask the most basic question of all in machine learning: how do we define the problem of machine learning? A Transformer is essentially a nonlinear approximator of sequence-to-sequence functions. It works because it is possible to train a Transformer model using an enormous amount of data, specifically trillions of tokens. But, more abstractly, suppose we are given a finite sample drawn from some set (e.g., in training a Transformer, we are given samples of vectors $x \in \mathbb{R}^{n \times d}$). How do we define the problem of extrapolating (or interpolating) the data over some finite subset of token sequences to the entire vector space? This problem unfortunately has no canonical solution. Much of the literature in machine learning and statistics over many decades has explored a wide range of approaches. [27]

[27] Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani. *An Introduction to Statistical Learning – with Applications in R*, volume 103 of *Springer Texts in Statistics*. Springer, New York, 2013. ISBN 978-1-4614-7137-0. DOI: 10.1007/DOI

PAC (Probably Approximately Correct) Learning is a theoretical framework that studies the computational complexity of learning from data [28]. In this framework, a teacher draws samples from some fixed distribution $\mathbb{P}$ over a space of data X, and presents it to a learner. In the setting of Transformer models, X is essentially the entire collection of digital data that is accessible (from books, blogs, social network postings etc.). The learner is expected to infer an approximation g to some desired function f such that for any given distribution $\mathbb{P}$, the error $\epsilon = \mathbb{P}(f(X) \neq g(x))$ is bounded. But this formulation does not specify any canonical solution, and there are many formalizations of machine learning. There is no formalization that is canonical. Ultimately, the fundamental problem of the goal inferring a function from data is ill-defined.

[28] Leslie G. Valiant. A theory of the learnable. *Commun. ACM*, 27(11):1134–1142, 1984. DOI: 10.1145/1968.1972. URL https://doi.org/10.1145/1968.1972

FUNCTORS remarkably change the problem from an inherently ill-defined problem into a well-defined problem. In this setting, one is given a functor $F : \mathcal{C} \to \mathcal{D}$, and the *extension* problem involves discovery of a new functor $G : \mathcal{E} \to \mathcal{D}$, given another functor $K : \mathcal{C} \to \mathcal{E}$. In other words, we are interested in *extending* the given functor F along K. To see the relationship between extending functors and extrapolating functions, imagine that the category $\mathcal{C}$ is a finite set of samples, and the category $\mathcal{E}$ is the larger (potentially infinite) category over which the function is to be extrapolated. Concretely, in the setting of Transformer models, $\mathcal{C}$ is the finite set of all possible token sequences that could be accessed during training, and $\mathcal{E}$ is the complete vector space $\mathbb{R}^{n \times d}$. Well, one of the most remarkable results in category theory, due to Kan, is that this functor extension problem has exactly two canonical solutions: the *left* Kan extension and the *right* Kan extension. [29]

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

Clustering as a Functor

One of the motifs that underlies many of the applications of category theory to AI and ML in this book and course is to model problems in terms of functors. The figure on the next page illustrates the use of functoriality in designing an algorithm for clustering points in a finite metric space based on pairwise distances, one of the most common ways to preprocess data in ML and statistics. Treating clustering as a functor implies the resulting algorithm should behave appropriately under suitable modifications of the input space. Here, we can define clustering formally as a functor F that maps the input category of finite metric spaces **FinMet** defined by (X, d), where X is a finite set of points in $\mathbb{R}^n$ and $d : X \times X \rightarrow [0, \infty]$ is a (generalized) finite metric space, and the output category **Part** is the set of all partitions X into subsets X_i such that $\cup_i X_i = X$.

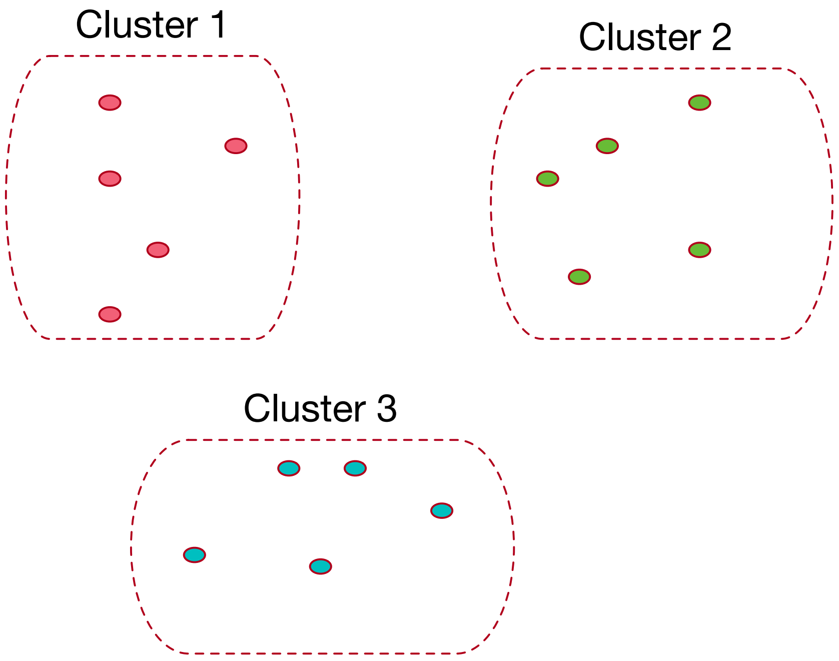


Figure 4: One of the most traditional problems in statistics and ML is *clustering* by constructing a partition of a finite metric space by grouping points together based on their pairwise distances. Treating clustering as a functor implies designing an algorithm that behaves functorially: if the distances were scaled by some factor, the clustering should not change.

One can impose three criteria on a clustering algorithm, which seem entirely natural, and yet, no standard clustering algorithm satisfies all these conditions. [30]

[30] Jon Kleinberg. An impossibility theorem for clustering. In S. Becker, S. Thrun, and K. Obermayer, editors, *Advances in Neural Information Processing Systems*, volume 15. MIT Press, 2002

- *Scale invariance:* If the distance metric d is increased or decreased by $c \cdot d$, where c is a scalar real number, the output clustering should not change. If the points in each cluster became closer together or further apart proportionally, the clustering should remain the same.
- *Completeness:* For any given partition of the space X, there should exist some distance function d such that the clustering algorithm when given

that distance function should return the desired partition.

- *Monotonicity:* If the distance between points within each cluster were decreased, and the distances between points in different clusters were increased, the clustering should not change either.

Remarkably, it turns out that no clustering algorithm exists that satisfies all these three basic conditions. Yet, even more remarkably, treating clustering as a functor makes it possible to define a modified clustering problem in terms of creating *persistent clusters* that overcomes Kleinberg's impossibility result. [31] This simple but deep example reveals the power of functorial design, and gives a concrete illustration of the importance of functorial thinking in AI and ML.

[31] Gunnar E. Carlsson and Facundo Mémoli. Classifying clustering schemes. *Found. Comput. Math.*, 13(2):221–252, 2013. DOI: 10.1007/s10208-012-9141-9. URL https://doi.org/10.1007/s10208-012-9141-9

Summary and Further Reading

This chapter introduced two of the most important ideas in categorical AGI: functors and natural transformations. It will take some time to appreciate these definitions, and the best way to do that is to go through as many examples as possible. Work through some sample exercises for yourself. [32]

[32] Given that RL algorithms can be modeled as functors, define a natural transformation between two RL algorithms.

UMAP (Uniform Manifold Approximation and Visualization) is the most powerful and efficient data visualization method in machine learning today. It is based on constructing functors from a high-dimensional dataset defined as a generalized metric space to a category of topological spaces. UMAP uses some advanced categorical structures, such as simplicial sets, which we will encounter later in this book and course. But it is worthwhile to explore UMAP through its downloadable `Python` package. [33]

[33] https://umap-learn.readthedocs.io/en/latest/how_umap_works.html

Representable Functors and the Yoneda Lemma

In this chapter, we will introduce one of the most elegant and powerful results in category theory: the Yoneda Lemma.[34] A story often told of its discovery by the "godfather" of category theory, Saunders Mac Lane, in the iconic Paris train station Gare du Nord, is quite entertaining to read.[35]

[34] A detailed discussion of the Yoneda Lemma is given in Chapter 2 of Riehl's textbook.

[35] http://www.neverendingbooks.org/le-lemme-de-la-gare-du-nord/

The Yoneda Lemma states simply that objects in a category can be defined purely on the basis of their interaction with all the other elements. It is one of those all-too-rare results in pure mathematics that can be described accurately in plain language. In other words, to understand an object x, simply examine all the influences $\mathcal{C}(-, x)$ that act on it, or equivalently, the objects it influences, $\mathcal{C}(x, -)$. The remarkable property that this Lemma reveals results in an embedding that has foundational implications for AGI, and in fact, can be seen as essential to the success of the Transformer model itself. To understand the Yoneda Lemma, we need to first understand how functors act on each other.

UNIVERSAL CAUSALITY is a categorical formulation of causality based on the Yoneda Lemma. [36] It leads to some deep insights into the problem of causal reasoning, because of a fundamental consequence of the Yoneda Lemma.

[36] Sridhar Mahadevan. Universal causality. *Entropy*, 25(4):574, 2023. DOI: 10.3390/E25040574. URL https://doi.org/10.3390/e25040574

Natural Transformations and Universal Arrows

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 11. *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:*

$$\begin{tikzcd}
Fc \arrow[r, "\alpha_c"] \arrow[d, "Ff"'] & Gc \arrow[d, "Gf"] \\
Fc' \arrow[r, "\alpha_{c'}"'] & Gc'
\end{tikzcd}$$

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 12. *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 $Sf' \circ u = f$.*

Once again, it is important to pause here, and try to understand what this definition is stating. [37]

[37] Draw the commutative diagram for this definition!

Definition 13. *If D is a category and $H : D \to$* **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 $(Hf)e = x$.*

A UNIVERSAL PROPERTY in category theory refers to a property that can be described by an initial or terminal object in a category of *diagrams* (which we will encounter in later chapters). When we use the phrase “universal" in category theory, we are referring to a very precise notion, namely a property that is definable in a unique way. Carefully read the section on universal representations in Riehl’s textbook. She uses an illustrative example of graph colorability. We know that not all graphs can be colored by n (e.g, 3) colors. But what is the universal property for graph colorability? [38]

[38] Try to define a universal property for reinforcement learning algorithms, by first defining universal arrows between the category $\mathcal{C}_{\text{MDP}}$ and the category $\mathcal{C}_V$ of value functions.

Theorem 1. *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.

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 $\hat{\mathcal{C}} = \mathbf{Set}^{\mathcal{C}^{op}}$. Remarkably, it forms a topos, and has many attractive properties that we will explore later in the book and course. The Yoneda lemma plays a crucial role in this book 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.

$$\begin{tikzcd}
D(r,r) \arrow[r, "\phi_r"] \arrow[d, "D(r,f')"] & C(c,Sr) \arrow[d, "C(c,Sf')"] \\
D(r,d) \arrow[r, "\phi_d"] & C(c,Sd)
\end{tikzcd}$$

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.

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*

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

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 14. *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.*

Yoneda Intuition via Visualizing Self-Attention in BERT

A SLOGAN that will guide this chapter is the following: *an object is determined by all of its relationships.* The Yoneda Lemma makes this precise by identifying an object $c \in C$ with the functor of morphisms out of (or into) it, such as $C(c, -)$ or $C(-, c)$, and by stating that natural transformations out of a representable functor are equivalent to elements of the target functor.

SELF-ATTENTION implements a remarkably similar principle in modern Transformer models. A token representation is not a fixed embedding, but is *context-sensitive*: the representation of a token depends on its learned relationships to other tokens in the sentence. This gives an empirical realization of a Yoneda-style idea: a token acquires its meaning-in-context through how it "sees" (and is seen by) other tokens.

ATTENTION AS AN ENRICHED REPRESENTABLE. Let A denote a token position in a sentence. In a toy causal "sequence category" (as in the Week 2 micro-lab), the representable functor $h_A(-) = \mathrm{Hom}(A, -)$ is the set of arrows from A to all other token positions. In a real Transformer, self-attention replaces a set-valued representable with a *weighted* (enriched) profile:

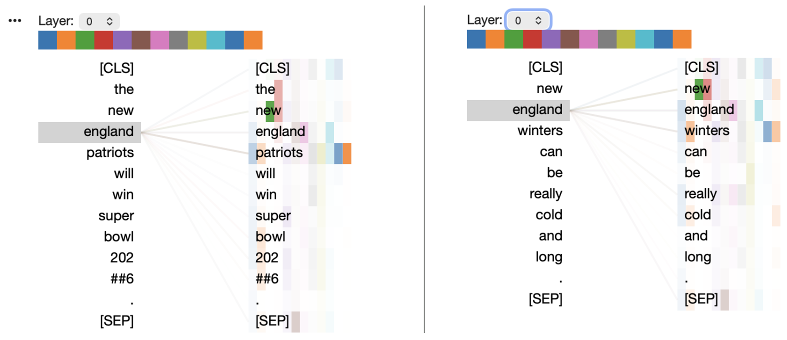
$$\alpha_{A\to j} = \mathrm{softmax}_j\Big(\frac{\langle q_A, k_j\rangle}{\sqrt{d}}\Big), \qquad y_A = \sum_j \alpha_{A\to j}\, v_j,$$

where q_A is the query vector for token A, k_j is the key vector for token j, and v_j is the value. Thus the updated representation y_A is computed by "evaluating" values along all relationships out of A, weighted by $\alpha_{A\to j}$.

A DISAMBIGUATION DEMO: "NEW ENGLAND". Consider the two sentences:

(1) *The New England Patriots will win Super Bowl 2026.*
(2) *New England winters can be really cold and long.*

The phrase *New England* is ambiguous: in (1) it refers primarily to a sports team brand/identity, while in (2) it refers to a geographic/climatic region. A pretrained model (e.g. BERT) resolves this ambiguity by producing different contextual encodings for the tokens *new* and *england*, which can be observed directly by visualizing attention weights. In (1), the attention profiles for *england* place mass on tokens such as *patriots*, *super*, *bowl*; in (2) they place mass on *winters*, *cold*, *long*. In other words, the "relationship profile" of the token changes with context, and the token's representation changes accordingly.



(a) Sports context (b) Weather context

Figure 5: **Attention as an enriched Yoneda profile.** Clicking the token *england* reveals its head-wise attention distribution $\alpha_{A\to -}$ at a fixed layer. The attention profile—and therefore the contextual embedding—changes with context, disambiguating the phrase *New England*.

VISUALIZATION. The figure above shows an interactive visualization of attention heads in BERT (using BertViz), highlighting the attention relations of *new* and *england* in the two contexts. A companion Colab notebook reproducing this visualization is provided in the course repository.

CONNECTION BACK TO YONEDA. The Yoneda Lemma states that an object c is determined by the functor $C(c, -)$ (or $C(-, c)$). Self-attention can be viewed as learning a *soft/enriched* analogue: a token representation is updated from a weighted profile of its relationships to all other tokens. In subsequent chapters, we will formalize how diagrammatic constraints (e.g. commutative squares) can be used to regularize such relationship profiles and how sheaf/topos structure can be used to glue local views into globally coherent commitments.

Summary and Further Reading

In this chapter, we introduced one of the most beautiful results in pure mathematics, the Yoneda Lemma. It states simply that objects can be defined purely on the basis of their interaction with other objects. It can be viewed without exaggeration as the foundation of modern digital marketing – “you are what you purchase online" – as well as social networking – “you are defined by your friends on a social network". There are many deep consequences of the Yoneda Lemma that we will encounter later in this book and course, which include new ways of modeling causal inference and deep learning.

An alternate form of the Yoneda Lemma makes it somewhat easier to appreciate:

$$\mathrm{Nat}(\mathcal{C}(-, c), \mathcal{C}(-, d)) \simeq \mathcal{C}(c, d)$$

If you think about causal inference in this context, what this result states is that the causal influences of a variable c upon another variable d is computable purely as a function of their associated presheaf objects $\mathcal{C}(-, c))$ and $\mathcal{C}(-, d)$.

Diagrams and Universal Constructions

We now move to a more difficult topic for most beginners learning category theory: universal constructions, such as *colimits* and *limits*. These concepts can initially be daunting, but fortunately when understood in the context of AGI applications, they turn out to be surprisingly useful. Fundamentally, they give us new ways to combine Transformer models or causal models, and allow constructing an assembly factory of novel representations from basic building blocks. Recall that an LLM is abstractly modeled as a sequence-to-sequence function. These become the basic objects in a topos category, which then can be shown to possess as (co)limits. As we will see, one consequence is that any “diagram" comprising of LLM objects as a “solution".

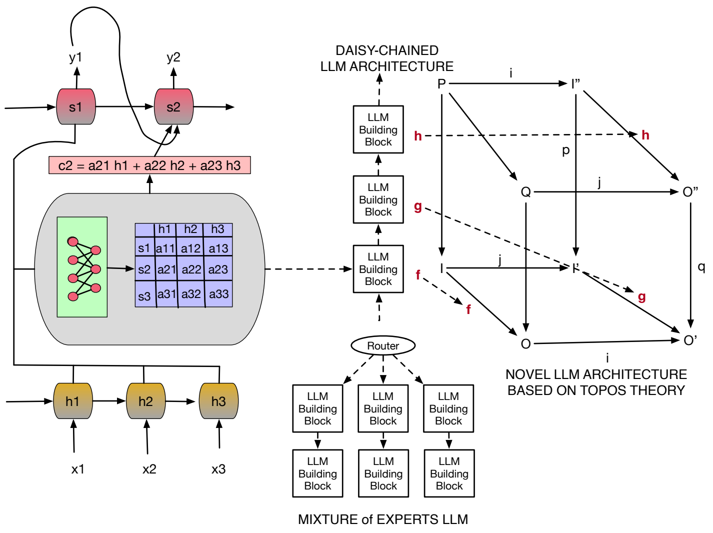


Figure 6: Limits and Colimits give us novel construction techniques for building LLM architectures.

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.[39] These concepts are among the most difficult to understand initially, and it will require much concentration to absorb their implications. As with all beautiful abstractions in pure mathematics, a deeper understanding of these ideas will reward the reader with a much deeper appreciation for their

[39] Read Chapter 3 of Riehl’s textbook for a great introduction to these concepts.

application to AGI.

DIAGRAMS are functors in disguise from some *indexing* category (usually finite) to an actual concrete category of interest. Let us consider as a simple example the concept of a *Cartesian* product of sets, which is usually defined as

$$A \times B = \{(a, b) | a \in A, b \in B\}$$

Consider now defining this same concept, except without "looking inside" an object. Given two abstract objects, say c and d, what is their "Cartesian product"? To work this out, we need to abstract out the definition of Cartesian product of sets in terms of the Yoneda Lemma. The "product object" is defined uniquely by its interaction with other members of the category. What are those? In effect, we are asking to define the Cartesian product of two sets, not by listing all possible ordered pairs, but how the Cartesian product maps onto every possible set! This seems like a hopeless task, but remarkably it turns out to be actually quite simple.

Definition 15. *A* diagram *$F : \mathcal{J} \rightarrow \mathcal{C}$ is a functor F from some finite category $\mathcal{J}$ into a category of interest, $\mathcal{C}$.*

For example, $\mathcal{J} = \bullet \rightarrow \bullet \leftarrow \bullet$ is an example of a "pullback" diagram. Here the $\bullet$ refer to abstract objects that are mapped into concrete objects in $\mathcal{C}$ by the functor F. What we want to know whether a particular diagram F or an entire class of diagrams is "solvable". What this means is whether its limit or colimit exists, that is, is the category *complete* or *co-complete*? We will see that these are indeed very useful properties in designing AGI systems.

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

$$Colimit_{\mathcal{D}} F = \bigsqcup_D \mathcal{F}(D)$$

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 6. *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*

$$limit_{\mathcal{D}} F = \prod_{D} \mathcal{F}(D)$$

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. We first begin with the pushforward construction, which can be seen as "gluing" things together.

$$\begin{tikzcd}
Z \arrow[r, "p"] \arrow[d, "q"] & X \arrow[d, "f"] \arrow[ddr, bend left, "h"] & \\
Y \arrow[r, "g"] \arrow[drr, bend right, "i"] & X \sqcup Y \arrow[dr, "r"] & \\
& & R
\end{tikzcd}$$

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 $i = r \circ g$. Coproducts are themselves special cases of the more general notion of colimits.

The next figure 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.

$$\begin{tikzcd}
T \arrow[drr, bend left, "x"] \arrow[ddr, bend right, "y"'] \arrow[dr, dotted, "k"] & & \\
& U \arrow[r, "g'"] \arrow[d, "f'"] & X \arrow[d, "f"] \\
& Y \arrow[r, "g"] & Z
\end{tikzcd}
\qquad
\begin{tikzcd}
T \arrow[drr, bend left, "x"] \arrow[ddr, bend right, "y"'] \arrow[dr, dotted, "k"] & & \\
& \int \delta \arrow[r, "\delta'"] \arrow[d, "\pi_\delta"] & \mathbf{Set}_* \arrow[d, "\pi"] \\
& S \arrow[r, "\delta"] & \mathbf{Set}
\end{tikzcd}$$

Figure 7: (**Left**) Universal Property of pullback mappings. (**Right**) The Grothendieck category of elements $\int \delta$ of any set-valued functor $\delta : S \to$ **Set** can be described as a pullback in the diagram of categories. Here, **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.

PULLBACK AND PUSHFORWARD diagrams are notable in terms of what they reveal: a pushforward is characterized by the maps going *into* it, whereas a *pullback* is characterized by the maps going *out* of it. Once again, remind yourself of the powerful Yoneda Lemma, which gives these definitions a precise theoretical grounding, for it states that any object (e.g. a pushforward or pullback) can be precisely characterized by either the maps going into it or the maps going out of it.

For any set-valued functor $\delta : S \to$ **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. [40]

[40] Read the properties of the Grothendieck category of elements in Riehl's textbook, especially their relationship to the Yoneda Lemma.

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 book 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 16. *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 $(\lambda_j : F_j \to c)_{j \in J}$.*

$$\begin{tikzcd}
& c \arrow[dl, "\lambda_j"'] \arrow[dr, "\lambda_k"] & \\
Fj \arrow[rr, "Ff"] & & Fk
\end{tikzcd}
\qquad
\begin{tikzcd}
Fj \arrow[rr, "Ff"] \arrow[dr, "\lambda_j"'] & & Fk \arrow[dl, "\lambda_k"] \\
& c &
\end{tikzcd}$$

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

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

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*

$$C(-, \lim F) \simeq Cone(-, F)$$

Dually, for colimits, we can define a functor

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

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

$$C(colimF, -) \simeq Cone(F, -)$$

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 18. *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 19. *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 $f \in$* **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:*

$$\begin{tikzcd}
 & F(D) \arrow[dl, "F(g)"'] \arrow[dr, "f"] & \\
F(D') \arrow[rr, "f'"] & & C
\end{tikzcd}$$

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

Definition 20. *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 $Colimit_{F \downarrow C} F \circ U \simeq C$. 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 2. 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.*

Universal Arrows and 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 21. *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 7. *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*.*

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 GAIA architecture.

Definition 22. *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* $\int \delta \in \mathbf{Cat}$ *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:*

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

Summary and Further Reading

This chapter has introduced some of the most abstract concepts you will study in this course, and we will explore its practical consequences in designing a new framework for deep learning in the next few chapters, as well as new architectures for Transformer models. So, rest assured, you will find plenty of concrete examples of these abstractions that will help you understand these ideas better.

Riehl's textbook has a detailed study of (co)limits, including a surprising proof that in the category of **Sets**, all diagrams are solvable, meaning that we can find the limit or colimit of any diagram. This result has profound implications for AGI, as we can now think of designing novel Transformer architectures based on these universal constructions. To help you visualize these ideas, imagine constructing a diagram where every object is a Transformer model, and every arrow is a commutative diagram between two Transformer models. Then, the (co)completeness of the category of **Sets** implies that any such diagram must be solvable. In other words, there exists some Transformer model that most closely "approximates" the diagram in the sense of natural transformations that come into or out of it. Of course, this theoretical result does not immediately imply that in practice, such a Transformer model can be easily constructed, for Transformer models must be trained from data. We will see in the next chapter how we can turn the theoretical concepts of (co)limits into actual practical deep learning algorithms.

Categorical Deep Learning

We now build on the powerful notions we have introduced thus far in previous chapters to design a new framework for deep learning in the following chapters. First, we review the basic idea of modeling deep learning as a functor in this chapter, and then introduce a powerful generalization of deep learning in the next chapter.

In Chapter 1, we saw that Transformer models could be formalized as a topos category, where each object is a Transformer model. In this chapter, we delve deeper into the Transformer model, in particular examining the core construct of *self-attention*. To understand how to construct a more fundamental categorical model for Transformers, we need to introduce more refined notions of categories, such as *symmetric monoidal categories* [41]. Let us how revisit the application of category theory to model not just the Transformer category, but to model large language models (LLMs) more generally as well. to model LLMs in this section. In the case of LLMs, as we will see below, a natural way to define objects is as “tokens" (roughly interpreted to mean words in a natural language). Naturally, words can be concatenated into sentences, which leads us to using *symmetric monoidal categories*.

[41] Brendan Fong and David I Spivak. *Seven Sketches in Compositionality: An Invitation to Applied Category Theory*. Cambridge University Press, 2018

Symmetric Monoidal Category

We now refine our notion of a category to include taking products of objects, where $c \otimes d$ is interpreted as a tensor product of objects c and d in some category $\mathcal{C}$. To interpret this concretely in AGI applications, such as Transformers, consider each object to be a token, or a sequence of tokens. The tensor product $\otimes$ can be interpreted as concatenation of two tokens.

Symmetric Monoidal Categories

Definition 23. *A* **monoidal category** *is a category C together with a functor* $\otimes : \mathcal{C} \times \mathcal{C} \to \mathcal{C}$*, an identity object e of C and natural isomorphisms* α, λ, ρ *defined as:*

$$\begin{aligned}\alpha_{C_1,C_2,C_3} : C_1 \otimes (C_2 \otimes C_3) &\cong (C_1 \otimes C_2) \otimes C_2, \\ \lambda_C : e \otimes C &\cong C, \\ \rho : C \otimes e &\cong C,\end{aligned}$$

The natural isomorphisms must satisfy coherence conditions called the “pentagon" and “triangle" diagrams. An important result shown in [42] is that these coherence conditions guarantee that all well-formed diagrams must commute. There are many natural examples of monoidal categories, the simplest one being the category of finite sets, where each object C is a set, and the tensor product $\otimes$ is the Cartesian product of sets, with functions acting as arrows. Other examples include the category of sets with relations as morphisms, and the category of Hilbert spaces. Markov categories are monoidal categories, where the identity element e is also a terminal object, meaning there is a unique “delete" morphism $d_e : X \rightarrow e$ associated with each object X. Recent work has shown that these form a unifying foundation for probabilistic and statistical reasoning [43].

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

[43] Tobias Fritz. A synthetic approach to markov kernels, conditional independence and theorems on sufficient statistics. *Advances in Mathematics*, 370: 107239, August 2020. ISSN 0001-8708. DOI: 10.1016/j.aim.2020.107239. URL http://dx.doi.org/10.1016/j.aim.2020.107239

Definition 24. *A* **symmetric monoidal category** *is a monoidal category* $(\mathcal{C}, \otimes, e, \alpha, \lambda, \rho)$ *together with a natural isomorphism*

$$\tau_{C_1,C_2} : C_1 \otimes C_2 \cong C_2 \otimes C_1, \text{ for all objects } C_1, C_2$$

where τ *satisfies the additional conditions: for all objects* C_1, C_2 $\tau_{C_2,C_1} \circ \tau_{C_1,C_2} \cong 1_{C_1 \otimes C_2}$, *and for all objects* C, $\rho_C = \lambda_C \circ \tau_{C,e} : C \otimes e \cong C$.

An additional hexagon axiom is required to ensure that the τ natural isomorphism is compatible with α. The τ operator is called a “swap" in Markov categories [44]. In most cases of interest in AI, the symmetric monoidal categories are *enriched* over some convenient base category $\mathcal{V}$, including vector spaces, or preorders such as the unit interval $[0, 1]$, where the unique morphism from $a \rightarrow b$ exists if and only if $a \leq b$. 5

[44] Tobias Fritz. A synthetic approach to markov kernels, conditional independence and theorems on sufficient statistics. *Advances in Mathematics*, 370: 107239, August 2020. ISSN 0001-8708. DOI: 10.1016/j.aim.2020.107239. URL http://dx.doi.org/10.1016/j.aim.2020.107239

Definition 25. *A* **V-enriched category** *consists of a regular category* $\mathcal{C}$, *such that for each pair of objects* x *and* y *in* $\mathcal{C}$, *the morphisms* $\mathcal{C}(x, y) \in \mathcal{V}$, *often referred to as a* $\mathcal{V}$*-hom object. For the case when* $(calV, \leq, \otimes, 1)$ *is a commutative monoidal preorder, we have the following conditions*

- $1 \leq \mathcal{C}(x, x)$
- $\mathcal{C}(y, z) \otimes \mathcal{C}(x, y) \leq \mathcal{C}(x, z)$

Attention in Transformers

Central to the structure of Transformer is the computation of attention scores. A detailed review of some types of attention scores is given in [45]. Ignoring many details, the overall idea is to compute some type of “dot product"

[45] Sneha Chaudhari, Varun Mithal, Gungor Polatkan, and Rohan Ramanath. An attentive survey of attention models, 2021. URL https://arxiv.org/abs/1904.02874

between two tokens that are embedded in Euclidean space. We provide a brief review of some approaches, and then define a category of Transformer models. To extend this basic symmetric computation of attention scores to nonsymmetric natural language applications, the inputs are weighted in some fashion such as using a sinusoidal function in the original Transformer model.

The general structure of an attention model is given as

$$A(q, K, v) = \sum_i p(a(k_i, q)) \times v_i$$

where K and V are key-value pairs, q is a query, a is an alignment function, and p is a distribution function. Some sample alignment functions are given in the following table.

Function	Equation
similarity	$a(k_i, q) = sim(k_i, q)$
dot product	$a(k_i, q) = q^T k_i$
scaled dot product	$a(k_i, q) = \frac{q^T k_i}{\sqrt{d_k}}$
general	$a(k_i, q) = q^T W k_i$
biased general	$a(k_i, q) = k_i(Wq + b)$
activated general	$a(k_i, q) = act(q^T W k_i + b)$
generalized kernel	$a(k_i, q) = \phi(q)^T \phi(k_i)$

Table 1: Some Alignment Functions used in Transformer models.

Transformers as a Category

To help concretize the more abstract presentation in the main paper, let us define a category of Transformer models. As with all generative AI systems, the fundamental structure of a Transformer model 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, 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 $[n] = \{1, \ldots, n\}$. 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 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 Transformer models compute specialized types of permutation-equivariant functions as defined by the commutative diagram below.

$$\begin{tikzcd}
X \arrow[r, "f"] \arrow[d, "P"'] & Y \arrow[r, "g"] \arrow[d, "P"] & Z \arrow[d, "P"] \\
XP \arrow[r, "f"'] & YP \arrow[r, "g"'] & ZP
\end{tikzcd}$$

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. Transformer models are inherently compositional, which makes them particularly convenient to model using category theory.

Definition 26. *A* **Transformer** *block is a sequence-to-sequence function mapping $\mathbb{R}^{d \times n} \rightarrow \mathbb{R}^{d \times n}$. There are generally two layers: a* self-attention *layer 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 $t^{h,m,r} : \mathbb{R}^{d \times n} : \mathbb{R}^{d \times n}$:*

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

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 $f : \mathbb{R}^{d \times n} \rightarrow \mathbb{R}^{d \times n}$ such that $f(XP) = f(X)P$ for any permutation matrix.

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-equivariant* 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 27. *The category $\mathcal{C}_T$ of Transformer models:*

- *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:*

$$T^{h,m,r} := \{f : \mathbb{R}^{d \times n} \to \mathbb{R}^{d \times n} \, | f(XP) = f(X)P\}$$

where P is some permutation matrix, h is the number of heads of size m each, and each feedforward layer has r hidden nodes.

LLM Next-token Distributions as Categories

Now, we define categories for LLMs based on next-token distributions, and introduce a hybrid category based on the k-NN LLM model proposed by [46].

[46] Urvashi Khandelwal, Omer Levy, Dan Jurafsky, Luke Zettlemoyer, and Mike Lewis. Generalization through memorization: Nearest neighbor language models, 2020. URL https://arxiv.org/abs/1911.00172

Definition 28. *The* **LLM syntax category** *$\mathcal{L}$ of a large language model is defined as a category enriched over a monoidal preorder $[0,1]$, whose objects are strings in a natural language, and whose morphisms are defined as $\mathcal{L}(x,y) := P(y|x)$, which means that if y is an expression such as "I am flying to San Diego", which extends the expression "I am flying", then $P(y|x)$ gives the conditional probability of such an extension (modulo a training set over which the model was trained).*

To define more "semantic" categories for LLMs, we use *enriched co-presheaves*: these are Yoneda embeddings where a token x is mapped into the set of all possible completions $\mathcal{L}(x,-)$ – the copresheaf – enriched over the unit interval $[0,1]$. The Yoneda embedding maps an object x to a covariant set-valued $\mathcal{L}(x,-)$ in general, but in special cases, such as for LLMs, this functor is itself an object of an enriched category, such as a symmetric monoidal preorder over the unit interval $[0,1]$. We can consider the Yoneda embedding of the LLM category $\mathcal{L}$ to be a "semantic" category that defines the meaning of sentences.

Definition 29. *For the LLM category $\mathcal{L}$ in Definition 28, the* **semantic category** *$\hat{\mathcal{L}} := [0,1]^{\mathcal{L}}$ is the $[0,1]$-enriched category of $[0,1]$-enriched co-presheaves on the $[0,1]$-category $\mathcal{L}$.*

Finally, let us define a new type of category based on the k-NN large language model.

Definition 30. *The* **k-NN LLM syntax category** *$\mathcal{L}_{kNN}$ is defined as a category whose morphisms $\mathcal{L}_{kNN}(y|x)$ are based on combining a synthetic*

k-NN computed probability from a datastore $\mathcal{D}$ and the LLM learned next-token distribution, defined as:

$$p(y|x) = \lambda p_{KNN}(y|x) + (1-\lambda)p_{LM}(y|x)$$

computed by querying a datastore $\langle \mathcal{K}, \mathcal{V} \rangle$ with an internal self-attention function computed by the Transformer model $f(x)$ to retrieve its k nearest neighbors $\mathcal{N}$ using some distance function $d(.,.)$, and using a softmax of their negative distances:

$$p_{KNN}(y|x) \propto \sum_{\langle k_i, v_i \rangle \in \mathcal{N}} \mathbf{1}_{y=v_i} \exp\left(-d(k_i, f(x))\right)$$

The k**NN LLM semantics category** *is then $\hat{\mathcal{L}}_{kNN} := [0,1]^{\mathcal{L}_{kNN}}$ is the $[0,1]$-enriched category of $[0,1]$-enriched copresheaves on the $[0,1]$-category $\mathcal{L}_{kNN}$.*

It turns out that k-NN LLMs model more accurately the next-token distribution over equivalent phrases, such as `Charles Dickens wrote` and `Charles Dickens is the author of`. A common method to fine-tune an LLM is to use reinforcement learning with human feedback (RLHF), which is now widely used in many deployed systems. We can model RLHF categorically by defining a category $\mathcal{C}_{RLHF}$, where each object m is an LLM, modeled say as above by the next-token distribution, and the arrow $f : m \to n$ exists if the LLM object m can be modified into an LLM object n using RLHF. Category theory allows an arbitrary level of compositionality, which makes this rich variety of categorical models possible.

Synthetic Token Probabilities using Nearest-Neighbor Language Modeling

k-NN nearest neighbor language models combine a *synthetic probability* constructed from key-query value pairs $\langle k_i v_i \rangle$ from each training example $\langle c_i, w_i \rangle$, where c_i is a *context-phrase*, such as `Charles Dickens wrote`, and $f(.)$ is a function that maps a context c to a fixed-length vector representation computed by a pre-trained Transformer model. For example, $f(c)$ could map c to a fixed-length vector output by some self-attention layer (which we described in the previous section). The datastore $\langle \mathcal{K}, \mathcal{V} \rangle$ is the set of all key-value pairs constructed from all the training examples in $\mathcal{D}$:

$$\langle \mathcal{K}, \mathcal{V} \rangle = \{(f(c_i), w_i) | (c_i, w_i) \in \mathcal{D}\}$$

At the time of testing, the language model learned by an LLM generates the output distribution $p_{LM}(y|x)$, and queries the datastore $\langle \mathcal{K}, \mathcal{V} \rangle$ with $f(x)$ to retrieve its k nearest neighbors $\mathcal{N}$ using some distance function $d(.,.)$. A synthetic probability is then computed over the neighbors using a softmax of their negative distances:

$$p_{KNN}(y|x) \propto \sum_{\langle k_i, v_i \rangle \in \mathcal{N}} \mathbf{1}_{y=v_i} \exp\left(-d(k_i, f(x))\right)$$

The final distribution for the k-NN LLM is then given by

$$p(y|x) = \lambda p_{KNN}(y|x) + (1-\lambda)p_{LM}(y|x)$$

Thus, we can define a new LLM category for k-NN LLMs in this way called $\mathcal{L}_{knn}$ where the next-token probabilities are a linear interpolation of the learned probabilities combined with a nearest-neighbor prediction.

Backpropagation as a Functor: Compositional Learning

Our principal goal in this section to review the categorical framework for deep learning proposed in [47], 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.

[47] Brendan Fong, David I. Spivak, and Rémy 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

Category of Supervised Learning

Functors can be used to 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 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.

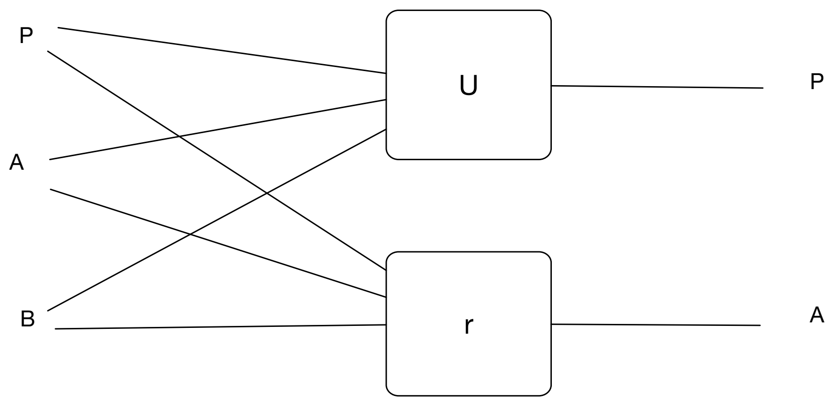


Figure 8: A learner in the symmetric monoidal category `Learn` is defined as a morphism.

Definition 31. *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.*

- *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 $(a, f(a)) \in A \times B$.

Note each learner can be combined in sequentially and in parallel, 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

$$(I \cdot J)(p, q, a) := J(q, I(p, a))$$

and the composite update function is

$$U \cdot V(p, q, a, c) := (U(p, a, s(q, I(p, a), c)), V(q, I(p, a), c)$$

and the composite request function is

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

Backpropagation as a Functor

We can define the backpropagation procedure as a functor that maps from the category `Para` to the category `Learn`. 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.

$$\begin{tikzcd} NNet \arrow[rr] \arrow[dr, "F"'] & & Learn \\ & Param \arrow[ur, "L_{\epsilon,e}"'] & \end{tikzcd}$$

Definition 32. *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)

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

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

where $E_I(p, a, b) := \sum_j e(I_j(p, a), b_j)$ 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 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.

Summary and Further Reading

In this chapter, we showed how deep learning, in particular backpropagation, can be theoretically viewed as a functor. In the next chapter, we turn theory into practice, and introduce an actual implementation that will build on a novel loss function constructed from categorical diagrams. While theoretically deep learning can be viewed as a functor, in practice, using backpropagation on actual neural networks does not automatically yield a functor as it is designed to only minimize associatively defined errors. Thus, we need to show how to make backpropagation truly functorial by introducing a new *curvature loss function*.

There are many extensions of the basic framework introduced here. For example, one can try to extend it to other types of machine learning. [48] We will see later that many other types of AGI applications, such as causal inference, can be studied using the principle of functoriality, combined with specific “nice" categories such as toposes.

[48] Design a functor for deep reinforcement learning, and think of why it might lead to a useful way to design novel deep RL methods.

Diagrammatic Backpropagation

In this chapter, we introduce a categorical framework for deep learning termed *Diagrammatic Backpropagation* (DB), and apply it to implement the *Geometric Transformer* (GT). DB minimizes the *curvature loss* of a categorical diagram to implement the GT theoretical conception of “horn filling" gaps in simplicial sets, a type of higher-order category. Crucially, DB exploits the property that in practice, neural net backpropagation is not a functor, and uses the deviation from functoriality as a loss function!

DB fundamentally builds on the concepts of (co)limits, so ensure that you have understood these universal constructions before reading this chapter, even if you are excited about learning a new type of backpropagation method, and want to jump directly into reading about DB. The whole design of DB was motivated by the goal of constructing a compositional learning framework that allows solutions to be “glued" together in a way that respects diagrammatic constraints. As we will show in this chapter and the next, “vanilla" backpropagation does not result in these diagrammatic constraints being respected, and we will formalize this failure into a loss function that can be optimized.

Introduction

In the previous chapter, we formulated backpropagation as a functor from a category of graphs to a category of compositional learners. We now show how to generalize this approach using simplicial sets –a type of higher-order category – as the domain of the backprop functor. [49] The principal novelty of DB is a novel loss function of minimizing the *diagrammatic curvature*. DB works by a process of “horn filling" in simplicial sets, which we sketch out algorithmically in this chapter. Simplicial sets are a combinatorial model of “nice" topological spaces, which generalize graphs. [50] In the next chapter, we will introduce a novel class of *Geometric Transformers* (GTs), which will use a type of “geometric" self-attention.

[49] Sridhar Mahadevan. GAIA: Categorical foundations of generative AI, 2024. URL https://arxiv.org/abs/2402.18732

[50] J.P. May. *Simplicial Objects in Algebraic Topology*. University of Chicago Press, 1992

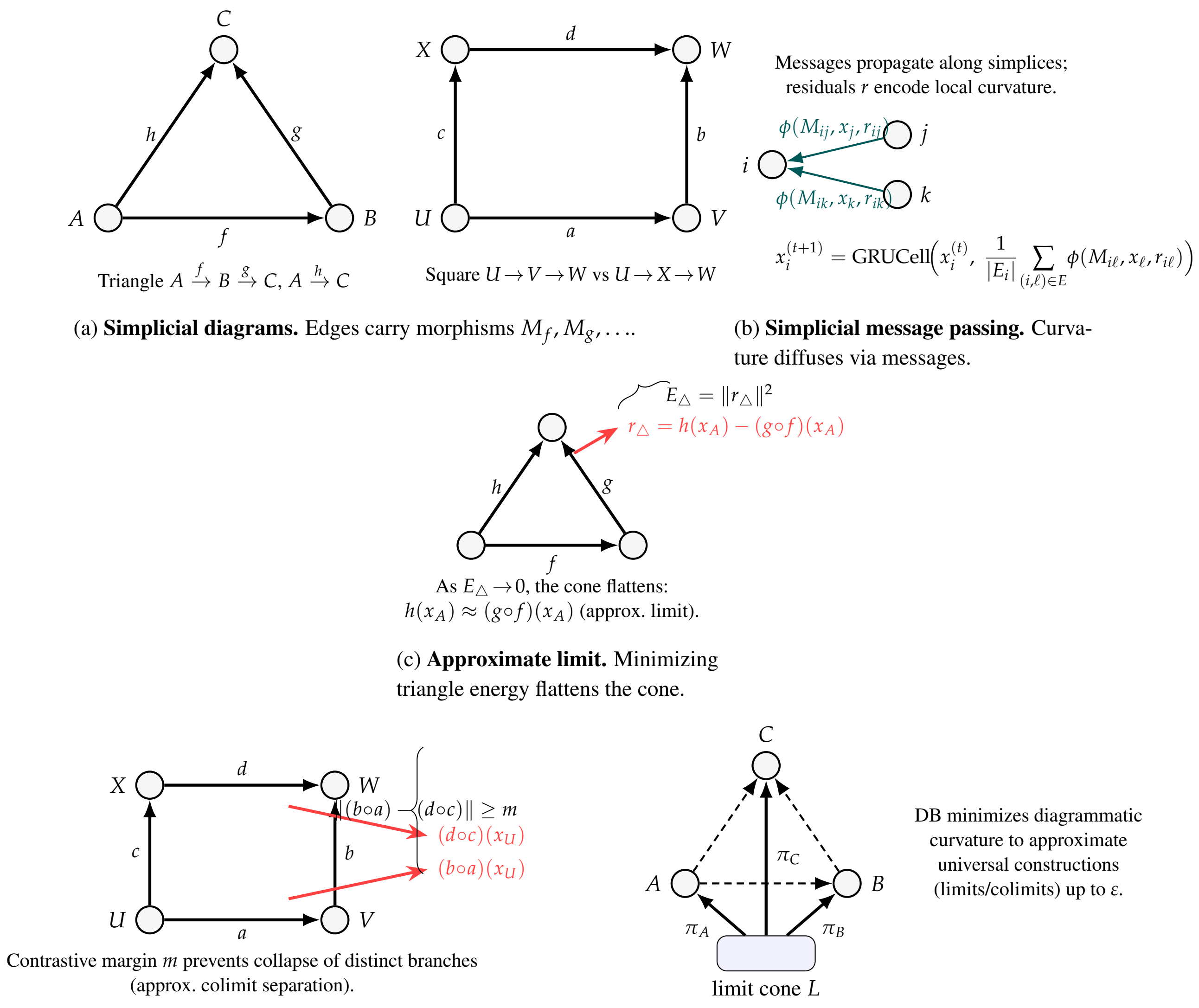
LLMs excel at natural language prediction but struggle with problems which test compositional algebraic reasoning. We hypothesize that this limitation arises from the lack of explicit functorial structure within their architectures. As part of a comprehensive set of experimental tests of DB and GT, we can show on simple benchmarks problems that are designed to test for compositional structure preservation, that the functorial design of DB reflects clearly in their superior performance.

Minimizing Diagrammatic Curvature Energy

DIAGRAMMATIC CURVATURE ENERGY is a novel loss function used in DB: it enforces algebraic constraints that must hold in any given task, and achieves superior performance over traditional backpropagation and Transformer approaches. In conventional supervised learning, training data are given as pairs (x, y), and a model learns a function $f_\theta(x)$ by minimizing a loss $\mathcal{L}(f_\theta(x), y)$ via backpropagation. This process enforces *local consistency* between predicted and target outputs, but treats each data point in isolation.

DB generalizes this paradigm by assuming that many learning problems possess an underlying *diagrammatic structure*—objects (data representations) connected by morphisms (transformations or relations) whose compositions obey algebraic constraints. A single diagram thus encodes not one example (x, y) but a small structured set of them linked by known functional relationships. Any category $\mathcal{C}$ can be converted into a *simplicial set*, which is a graded set X_n, where X_0 defines the objects, X_1 defines the 1-step arrows, and $X_n, n > 1$ defines higher-level simplices. For instance, in the Triangles benchmark, the objects A, B, C are latent feature vectors and the morphisms $f{:}A \to B$, $g{:}B \to C$, $h{:}A \to C$ represent linear maps expected to satisfy $h = g \circ f$. Instead of minimizing only per-sample prediction error, DB augments the loss with a *diagrammatic curvature term* that penalizes violations of such relations across the entire diagram. Backpropagation through this term propagates gradients not only along individual edges but around closed paths, enforcing global consistency. DB approximates universal constructions, such as (co)limits, which were introduced in Chapter 4.

ALGORITHM 1: SIMPLICIAL MESSAGE PASSING IN DB. DB propagates information through a simplicial complex representing a diagram of objects and morphisms. Each node corresponds to an object, and each directed edge corresponds to a morphism equipped with a learned linear map M_{ij} and relation embedding r_{ij}. At every iteration, a node aggregates messages arriving from its incident edges, each message combining the transformed neighbor representation $M_{ij}x_j$, a relation embedding, and any residual features that capture local curvature. The messages are degree-normalized and used to update the node state through a gated recurrent cell, allowing curvature information to diffuse across the entire diagram. Because each message includes a normalized edge transform, the updates remain numerically stable even when diagram curvature is large early in training. The process acts as


C
h
g
A
B
f
Triangle $A \xrightarrow{f} B \xrightarrow{g} C$, $A \xrightarrow{h} C$
X
d
W
c
b
U
a
V
Square $U \to V \to W$ vs $U \to X \to W$
Messages propagate along simplices; residuals r encode local curvature.
$\phi(M_{ij}, x_j, r_{ij})$
j
i
$\phi(M_{ik}, x_k, r_{ik})$
k
$x_i^{(t+1)} = \mathrm{GRUCell}\Big(x_i^{(t)}, \frac{1}{|E_i|} \sum_{(i,\ell)\in E} \phi(M_{i\ell}, x_\ell, r_{i\ell})\Big)$
$E_\triangle = \|r_\triangle\|^2$
$r_\triangle = h(x_A) - (g \circ f)(x_A)$
As $E_\triangle \to 0$, the cone flattens:
$h(x_A) \approx (g \circ f)(x_A)$ (approx. limit).
$\|(b \circ a) - (d \circ c)\| \geq m$
$(d \circ c)(x_U)$
$(b \circ a)(x_U)$
Contrastive margin m prevents collapse of distinct branches (approx. colimit separation).
π_C
π_A
π_B
limit cone L
DB minimizes diagrammatic curvature to approximate universal constructions (limits/colimits) up to ε.

(a) **Simplicial diagrams.** Edges carry morphisms $M_f, M_g, \ldots$.

(b) **Simplicial message passing.** Curvature diffuses via messages.

(c) **Approximate limit.** Minimizing triangle energy flattens the cone.

(d) **Approximate colimit.** Margin keeps branches distinct.

(e) **Universal view.** Curvature $\to 0 \Rightarrow$ numeric limit; margin $\Rightarrow$ numeric colimit.

Figure 9: **Simplicial message passing and limit/colimit approximation in DB.** (A) Triangle and square simplices encode compositional constraints. (B) DB aggregates messages along simplices; residuals carry local curvature. (C) Minimizing triangle energy $\|h(x_A) - (g \circ f)(x_A)\|^2$ flattens the cone (approx. limit). (D) A margin on square paths prevents collapse (approx. colimit).

a discrete diffusion of curvature error—propagating adjustments so that the node embeddings evolve toward a configuration in which all compositions of morphisms commute up to small residuals. In categorical terms, this iterative refinement approximates the flattening of a simplicial cone, pushing the learned functor F_θ closer to a limit object.

Algorithm 1: Simplicial Message Passing in DB

Input: Diagram $\mathcal{D} = (V, E)$ with objects V and morphisms E, node states $x_i^{(0)} \in \mathbb{R}^d$, edge maps $M_{ij} \in \mathbb{R}^{d\times d}$, relation ids r_{ij}, residual features ρ_{ij}, steps T

Output: Refined node states $x_i^{(T)}$

```
1:  for $t = 0, 1, \dots, T-1$ do
2:      for each node $i \in V$ do
3:          $m_i \leftarrow 0$
4:          for each incoming edge $(i \leftarrow j) \in E$ do
5:              $z \leftarrow M_{ij} x_j^{(t)}$;   $z \leftarrow z / (\|z\|_2 + \varepsilon)$        ▷ normalize edge transform
6:              $u \leftarrow \phi(z, r_{ij}, \rho_{ij})$   ▷ e.g. $u = \tanh(W_e[z;\rho] + b_e) + \mathrm{Emb}(r)$
7:              $m_i \leftarrow m_i + u$
8:          end for
9:          $m_i \leftarrow \frac{1}{\max(1, |\{j : (i \leftarrow j) \in E\}|)} m_i$        ▷ degree-normalize messages
10:         $x_i^{(t+1)} \leftarrow \mathrm{GRUCell}(x_i^{(t)}, m_i)$   ▷ curvature diffuses via messages
11:     end for
12: end for
```

ALGORITHM 2: LEARNED COMPOSITIONAL ENERGY. This routine computes the *diagrammatic curvature energy* $E_\theta(\mathcal{D})$ from the learned path embeddings. In the algorithm, ET_θ represents $\mathrm{edge\_transform}_\theta$. For each triangle constraint, the algorithm measures the discrepancy between the direct morphism h and the composed morphism $g \circ f$. For each square, it compares the two-path compositions $(b \circ a)$ and $(d \circ c)$. These residuals are squared and accumulated to yield $E_\triangle$ and $E_\square$. The resulting energy quantifies how far the learned representation is from exact commutativity: $E_\theta(\mathcal{D}) = 0$ if and only if all diagrams commute perfectly. Operationally, minimizing this energy aligns all composed morphisms in representation space, thereby reducing curvature. As training progresses and $E_\theta \to 0$, the geometry of the learned functor approximates a categorical *limit*, where multiple arrows converge to a single universal cone. This provides the numeric foundation for DB's interpretation as a “limit-seeking” optimizer.

ALGORITHM 3: RAW ENERGY VECTOR. The raw energy computation isolates a structural signature derived solely from the exogenous morphism

Algorithm 2: Learned Compositional Energy $E_\theta(\mathcal{D})$

```
Input: Triangle constraints $\mathcal{T}$, square constraints $\mathcal{S}$, node states $x$, maps $M$,
    relations $r$
Output: $E_\theta(\mathcal{D}) = E_\triangle + E_\square$
 1: $E_\triangle \leftarrow 0,\ E_\square \leftarrow 0$
 2: for each triangle $(A \xrightarrow{f} B \xrightarrow{g} C,\ A \xrightarrow{h} C) \in \mathcal{T}$ do
 3:     $z_h \leftarrow \mathrm{ET}_\theta(M_h, r_h, x_A)$
 4:     $z_{gf} \leftarrow \mathrm{ET}_\theta(M_g, r_g,\ \mathrm{ET}_\theta(M_f, r_f, x_A))$
 5:     $E_\triangle \leftarrow E_\triangle + \|z_h - z_{gf}\|_2^2$
 6: end for
 7: for each square $(U \xrightarrow{a} V \xrightarrow{b} W,\ U \xrightarrow{c} X \xrightarrow{d} W) \in \mathcal{S}$ do
 8:     $z_{ba} \leftarrow \mathrm{ET}_\theta(M_b, r_b,\ \mathrm{ET}_\theta(M_a, r_a, x_U))$
 9:     $z_{dc} \leftarrow \mathrm{ET}_\theta(M_d, r_d,\ \mathrm{ET}_\theta(M_c, r_c, x_U))$
10:     $E_\square \leftarrow E_\square + \|z_{ba} - z_{dc}\|_2^2$
11: end for
12: return $E_\triangle + E_\square$
```

matrices M_f. It measures the same triangle and square residuals as Algorithm 2 but without involving the learnable message-passing parameters. The result is a two-dimensional feature vector $[E_\triangle^{raw}, E_\square^{raw}]$ that serves as the input to a lightweight classifier head. Because these values are independent of the learned path, they form a stable, task-agnostic description of the diagram's inherent commutativity or inconsistency. During training, the head learns to map these energy magnitudes to class labels—e.g., "commuting" versus "corrupted." In this way, DB separates *structure discovery* (via E^{raw}) from *geometry enforcement* (via E_θ), achieving both interpretability and numerical stability.

Algorithm 3: Raw Energy Vector $E^{raw}(\mathcal{D}) = [E_\triangle^{raw}, E_\square^{raw}]$

```
Input: Constraints $(\mathcal{T}, \mathcal{S})$, node states $x$, exogenous maps $M$
Output: $[E_\triangle^{raw}, E_\square^{raw}]$
 1: $E_\triangle^{raw} \leftarrow 0,\ E_\square^{raw} \leftarrow 0$
 2: for each triangle $(f, g, h, A)$ in $\mathcal{T}$ do
 3:     $E_\triangle^{raw} \leftarrow E_\triangle^{raw} + \|M_h x_A - (M_g M_f) x_A\|_2^2$
 4: end for
 5: for each square $(a, b, c, d, U)$ in $\mathcal{S}$ do
 6:     $E_\square^{raw} \leftarrow E_\square^{raw} + \|M_b M_a - M_d M_c\|_F^2$
 7: end for
 8: return $\left[E_\triangle^{raw}/|\mathcal{T}|,\ E_\square^{raw}/|\mathcal{S}|\right]$
```

ALGORITHM 4: DB TRAINING LOOP. This is the unifying optimization

routine that integrates message passing, energy minimization, and contrastive margin control. For each batch of diagrams, DB first computes the raw and learned energies, then classifies each sample using the raw-energy head. The total loss combines a cross-entropy classification term with the geometric energy terms. Positive diagrams receive a direct curvature penalty proportional to E_θ, encouraging exact commutativity; negatives are trained with a contrastive hinge that enforces a minimum energy margin m, preventing degenerate flattening. A warm-up schedule for λ_{geom} allows the classifier to stabilize before the geometric term dominates, mirroring a gradual transition from syntactic to semantic consistency. The optimizer (AdamW with weight decay and gradient clipping) updates both the message-passing parameters and the head, while normalization of the edge transforms ensures bounded activations. Collectively, this loop drives DB toward *diagrammatic flatness*—numerically realizing categorical limits while preserving colimit-style diversity through the contrastive margin.

Algorithm 4: DB Training Loop

```
Input: Dataset $\{\mathcal{D}_i, y_i\}$ with diagrams and labels, steps $T$, warmup $T_w$,
    hyperparams $(\lambda_{\text{geom}}, \lambda_{\text{contr}}, m)$
Output: Parameters $\theta$ (message-passing + edge transforms) and head $\psi$
 1: for $t = 1, \ldots, T$ do
 2:     for minibatch $\mathcal{B}$ do
 3:         $\mathcal{L} \leftarrow 0$
 4:         for each $(\mathcal{D}, y) \in \mathcal{B}$ do
 5:             $E^{raw} \leftarrow$ Alg. 3;    $E_\theta \leftarrow$ Alg. 2
 6:             $\hat{y} \leftarrow \text{Head}_\psi(E^{raw})$          ▷ small MLP on $[E^{raw}_\triangle, E^{raw}_\square]$
 7:             $\mathcal{L} \leftarrow \mathcal{L} + \text{CE}(\hat{y}, y)$
 8:             $\lambda_g \leftarrow \lambda_{\text{geom}} \cdot \min(1, t/T_w)$          ▷ geometric warm-up
 9:             if $y = 1$ then          ▷ positives: approximate limits
10:                 $\mathcal{L} \leftarrow \mathcal{L} + \lambda_g E_\theta$
11:             else          ▷ negatives: enforce $\varepsilon$-colimit separation
12:                 $\mathcal{L} \leftarrow \mathcal{L} + \lambda_{\text{contr}} \cdot \max(0, m - E_\theta)^2$
13:             end if
14:         end for
15:         Update $(\theta, \psi)$ by AdamW + clipping;    normalize edge trans-
    forms
16:     end for
17: end for
```

ALGORITHM 5: CONSTRAINT EXTRACTION FROM DIAGRAM. This preprocessing step converts each diagram or algebraic problem instance into a structured set of triangle and square constraints. It scans all triples of connected morphisms to identify potential triangles $A \xrightarrow{f} B \xrightarrow{g} C$ with a direct shortcut $A \xrightarrow{h} C$, and all quadruples forming squares $U \xrightarrow{a} V \xrightarrow{b} W$ versus $U \xrightarrow{c} X \xrightarrow{d} W$. These constraint lists define the simplicial skeleton over which DB performs message passing and energy evaluation. By abstracting away the raw graph details, this encoding process provides a *functorial interface*: any problem that can be expressed as a commutative diagram—be it a symbolic algebra theorem, a logic-grid puzzle, or a dynamic-programming dependency—can be compiled into the same constraint form. This universality of representation is what allows DB to operate across domains, interpreting every structured reasoning task as the pursuit of diagrammatic curvature minimization.

```
Algorithm 5: Constraint Extraction from Diagram
Input: Diagram $\mathcal{D} = (V, E)$ with typed morphisms; anchor rule (e.g. choose $A$ in triangles)
Output: Triangle set $\mathcal{T}$, square set $\mathcal{S}$
1: $\mathcal{T} \leftarrow \emptyset$, $\mathcal{S} \leftarrow \emptyset$
2: for each triple $(A \xrightarrow{f} B \xrightarrow{g} C)$ with $A \xrightarrow{h} C$ do
3:     $\mathcal{T} \leftarrow \mathcal{T} \cup \{(f, g, h, A)\}$
4: end for
5: for each quadruple $(U \xrightarrow{a} V \xrightarrow{b} W,\ U \xrightarrow{c} X \xrightarrow{d} W)$ do
6:     $\mathcal{S} \leftarrow \mathcal{S} \cup \{(a, b, c, d, U)\}$
7: end for
8: return $(\mathcal{T}, \mathcal{S})$
```

Summary and Further Reading

In this chapter, we have taken our first big step in applying category theory to design a new framework for AGI: diagrammatic backpropagation (DB) is a novel compositional learning method that is designed to “glue" together individual solutions to small ML problems in a more reliable way than vanilla backpropagation can achieve. The real test of DB will come in the next chapter, when we introduce the Geometric Transformer (GT) model and show how it outperforms standard Transformer models on a wide range of language modeling tasks.

There are a number of related studies that generalize deep learning using categorical methods, and you can find references to this literature on the web. [51]

[51] Mustafa Hajij, Lennart Bastian, Sarah Osentoski, Hardik Kabaria, John L. Davenport, Sheik Dawood, Balaji Cherukuri, Joseph G. Kocheemoolayil, Nastaran Shahmansouri, Adrian Lew, Theodore Papamarkou, and Tolga Birdal. Copresheaf topological neural networks: A generalized deep learning framework, 2025. URL https://arxiv.org/abs/2505.21251

Geometric Transformers

In this chapter, we introduce a novel class of Geometric Transformers (GTs), implemented using the Diagrammatic Backpropagation (DB) framework introduced in the previous chapter. As we will show experimentally in this chapter, GT models dramatically outperform regular Transformers in data-limited regimes on complex datasets, such as the 100-million token `Wiki-103`. A Transformer can be regarded as an implementation of the Yoneda lemma on individual tokens, whereas the Geometric Transformer applies a 'path-space Yoneda' to short compositional neighborhoods. This extra structure appears to delay or avoid the information bottleneck that flattens the training curves of the baseline Transformer. We will introduce a *mean-field* theoretical analysis in the next chapter to show why GT models outperform traditional Transformers.

We thus take another significant step in the application of category theory to AGI in this chapter by designing a novel Transformer model, whose very architecture is motivated by the abstractions afforded by category theory. Its design could not have been accomplished without the fundamental concepts that we studied in previous chapters, such as (co)limits.

A Yoneda–style view of the Geometric Transformer

GEOMETRIC TRANSFORMER architectures are illustrated in the following page, and will be introduced in this section using the categorical machinery we have developed in the previous chapters. Let $\mathcal{C}$ be a small category whose objects are token–position pairs $(\text{token}, \text{position})$ in a sequence, and whose morphisms encode the basic "can attend to" relations (e.g., causal or bidirectional attention, positional shifts, etc.).

For each object $x \in \mathcal{C}$ the *representable presheaf*

$$h_x := \mathcal{C}(-,x) : \mathcal{C}^{\text{op}} \to \mathbf{Set}$$

assigns to each $x \in \mathcal{C}$ the set of morphisms $z \to x$. The Yoneda lemma says that

$$\mathcal{C}(x,y) \;\cong\; \text{Nat}\big(\mathcal{C}(-,x), \mathcal{C}(-,y)\big) \;=\; \text{Nat}(h_x, h_y), \qquad (1)$$

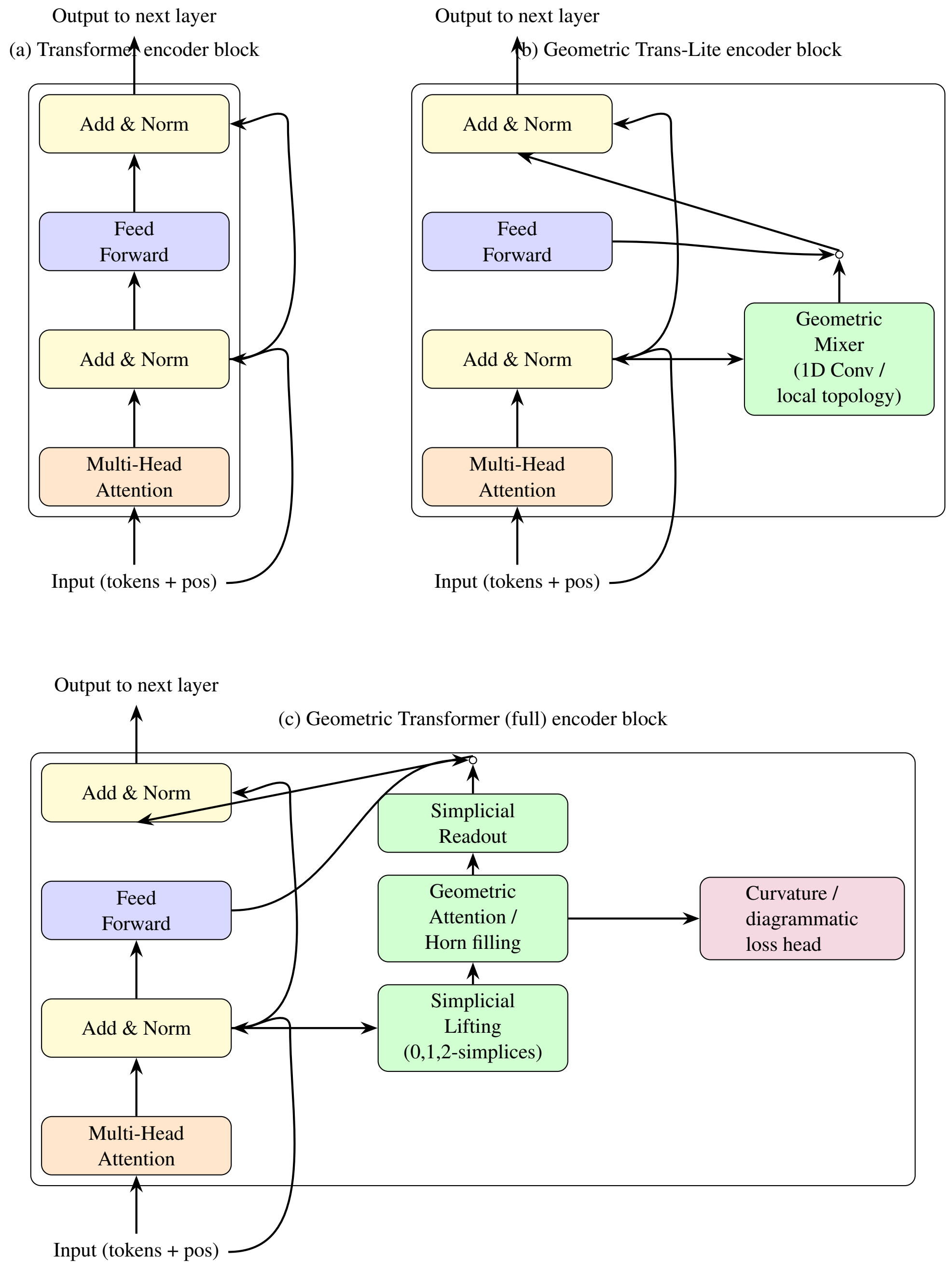


Figure 10: Comparison of encoder blocks. (a) Standard Transformer encoder block. (b) GeomTrans-Lite augments each layer with a local geometric mixer. (c) The full Geometric Transformer replaces the mixer with a simplicial lifting–mixing–readout stack and adds an auxiliary curvature / diagrammatic loss head, introducing an explicit geometric inductive bias.

i.e., a morphism $x \to y$ is equivalent to a natural transformation between the representables h_x and h_y.

A categorical interpretation of self–attention is:

- each token x carries a learned embedding that encodes (a parametrization of) the functor h_x,
- an attention head computes scores that approximate a natural transformation $h_x \Rightarrow h_y$ between such functors, and
- the weighted sum of value vectors is a finite / soft evaluation of this natural transformation on the current batch of objects.

In short, *vanilla self–attention is Yoneda with plain representables* h_x: the model only ever sees "all arrows into x", not the richer small diagrams those arrows can form.

Diagram functors for GT Lite and GT Full

The Geometric Transformer introduces two extra ingredients:

1. a *local geometric path*, implemented as a 1D convolution/smoothing over neighboring tokens (GT–Lite), and
2. a *simplicial / diagrammatic message passing* step, in which information flows not only along single edges, but also around small relational motifs (triangles, horns), together with a curvature term that penalizes inconsistency across such motifs (GT–Full).

Categorically, this can be expressed as replacing the bare representable h_x by a richer *diagram functor* that depends on the small diagram of arrows, paths, and 2–simplices around x.

PATH CATEGORY AND DIAGRAM FUNCTORS. Let $\mathbf{Path}(\mathcal{C})$ be the category whose objects are finite paths in $\mathcal{C}$,

$$z_0 \xrightarrow{f_1} z_1 \xrightarrow{f_2} \cdots \xrightarrow{f_n} x,$$

and whose morphisms are commuting maps between such paths. For each $x \in \mathcal{C}$, define a *diagram functor*

$$D_x : \mathbf{Path}(\mathcal{C})^{\mathrm{op}} \to \mathbf{Vect},$$

which assigns to each path $p : z_0 \to \cdots \to x$ a vector space (or, in practice, a single vector) and to each morphism of paths a linear map.

Intuitively, D_x does not just remember the arrows $z \to x$ (as h_x does); it remembers entire *local diagrams ending at* x: paths, triangles, and higher simplices. In the implementation:

- the 1D convolution in GT–Lite can be viewed as providing a *smoothed neighborhood* representation: instead of a single representable h_x, we work with a small aggregate

$$\tilde{h}_x \;\approx\; \int^{z\in N(x)} \mathcal{C}(-,z)$$

over neighbors $z \in N(x)$, i.e. an approximate coend over a local neighborhood;

- the simplicial message passing in GT–Full takes as input embeddings for edges and triangles and returns a refined embedding that depends on the entire 1– and 2–simplicial structure around x. This is a learned parametrization of the functor D_x on the nerve $N(\mathcal{C})$ (the simplicial set of composable chains in $\mathcal{C}$).

CURVATURE LOSS AS FUNCTORIALITY CONSTRAINT. The diagrammatic backpropagation (DB) loss penalizes violations of "flatness" across 2–simplices. Concretely, for any triangle

$$z \xrightarrow{g} x, \quad z \xrightarrow{h} y, \quad x \xrightarrow{k} y,$$

DB enforces a local consistency constraint on the embeddings assigned to z, x, y and to the composite paths $z \to x \to y$ and $z \to y$. In categorical terms, this encourages D_x and D_y to behave as honest functors on the nerve, so that parallel paths have nearly the same image in **Vect**; the residual is interpreted as a *discrete curvature*.

Thus, GT does not only learn a representation of *objects* via h_x, but a representation of the entire *diagram of paths and simplices ending at* x via D_x, with DB enforcing that D_x is almost flat as a functor on $N(\mathcal{C})$.

A diagrammatic Yoneda principle

With this notation in hand, we can state an informal "GT Yoneda principle".

In a vanilla Transformer, an attention head learns (a soft finite approximation of) a natural transformation

$$\mathsf{Attn}_{x,y} \;\in\; \mathrm{Nat}\big(h_x, h_y\big) = \mathrm{Nat}\big(\mathcal{C}(-,x), \mathcal{C}(-,y)\big),$$

in accordance with the classical Yoneda lemma (1).

In a Geometric Transformer, the same head instead learns a natural transformation between *diagram functors*

$$\mathsf{GeomAttn}_{x,y} \;\in\; \mathrm{Nat}\big(D_x, D_y\big),$$

where $D_x, D_y : \mathbf{Path}(\mathcal{C})^{\mathrm{op}} \to \mathbf{Vect}$ encode not only representable structure but the entire small simplicial neighborhood of x and y. The DB curvature loss encourages these D_x to be as functorial (flat) as possible on the nerve $N(\mathcal{C})$.

At a high level, the difference is:

Vanilla Transformer	**Geometric Transformer**
tokens $\mapsto$ representables h_x	tokens $\mapsto$ diagram functors D_x
attention $\approx \mathrm{Nat}(h_x, h_y)$	attention $\approx \mathrm{Nat}(D_x, D_y)$
no explicit higher–order structure	explicit use of paths / triangles and curvature

This suggests that the empirical gains of GT over vanilla Transformers on PTB / WikiText–103 (see Sec.) can be viewed categorically as replacing the coarse "Yoneda on points" view of tokens by a finer "Yoneda on local diagrams" view: the model is not only sensitive to which tokens attend to which, but also to how those relations compose into paths and 2–simplices, with DB acting as a discrete curvature regularizer on the resulting functors D_x.

Diagrammatic Backpropagation as Horn Filling in Simplicial Sets

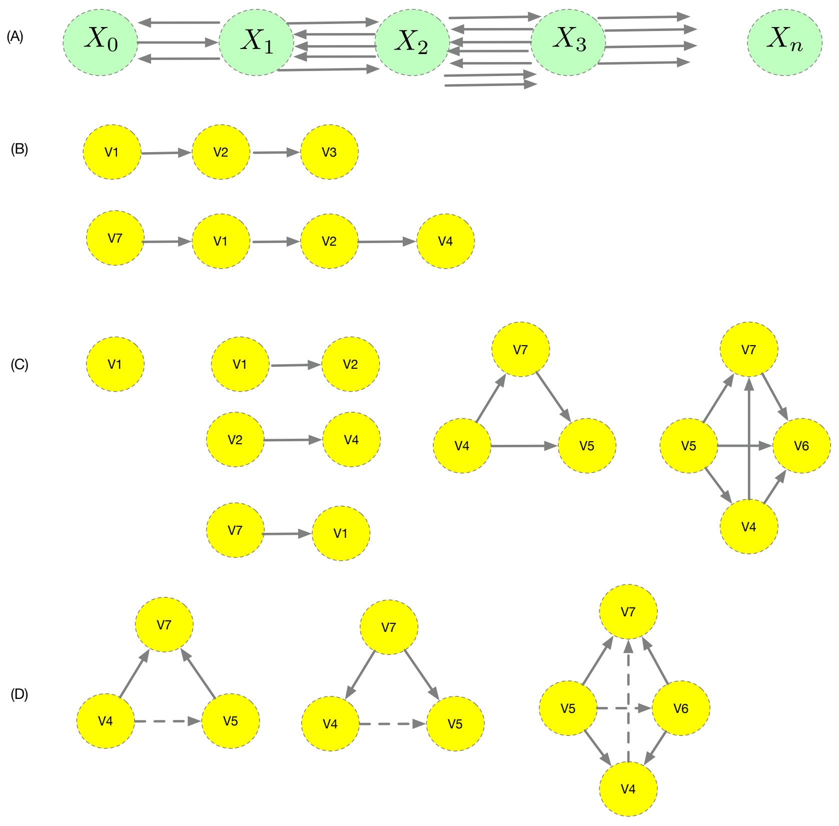


Figure 11: (A) DB and GT are based on *hierarchical* simplicial sets and objects. The base simplicial set X_0 is a set of entities that can be mapped to computational entities in generative AI, such as a tokens in a large language model, or images in a diffusion based system. The set X_1 defines a collection of morphisms between pairs of objects in X_0, where each morphism could define a deep learning module. (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

In this section, we give a theoretical explanation of DB's use of the framework of "horn filling" in a simplicial set. Unlike earlier generative AI architectures, GAIA uses the paradigm of simplicial sets and objects as the basic building blocks for generative AI. GAIA puts together building blocks of generative AI methods as n-simplices of 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.

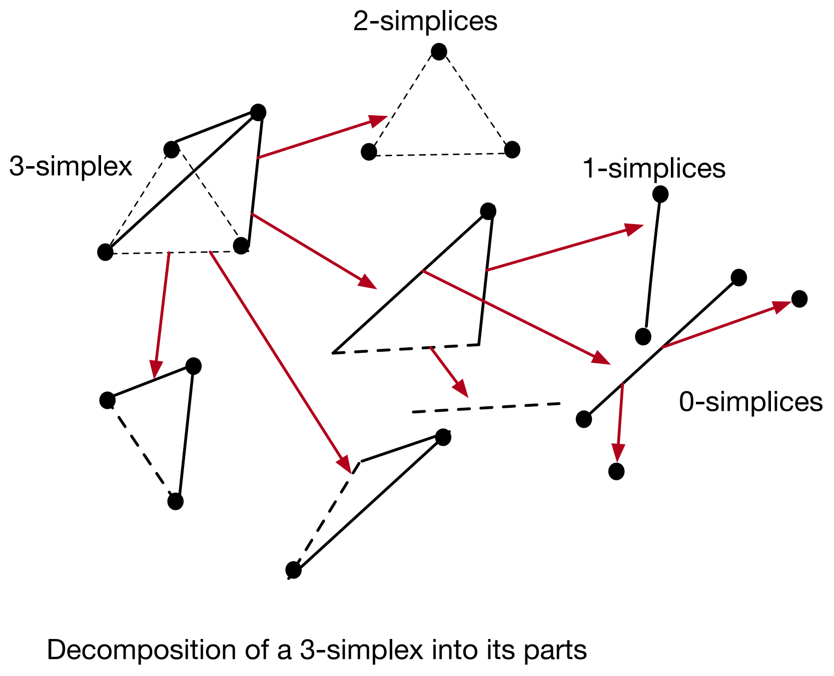


Figure 12: The *hierarchical* framework underlying DB and GT, where each n-simplicial complex acts as a business unit in a company: each n-simplex updates its parameters based on data it receives from its superiors, and it transmits guidelines for its $n + 1$ sub-simplicial complexes to help them with their updates. The mathematics for this hierarchical framework is based on higher-order category theory of simplicial sets and objects.

DB and GT use the simplicial category Δ 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 $[n], n \geq 0$, but really “comes to life" when it is plugged into some concrete category, such as the symmetric monoidal category of compositional learners `Learn` or vector spaces `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. We can also define functors from Δ into some category of generative AI models.

Simplicial Sets and Objects

A simplicial set can be viewed as a collection of sets, or a *graded* set, $S_n, n \geq 0$, where S_0 defines the primitive objects (which can be elements

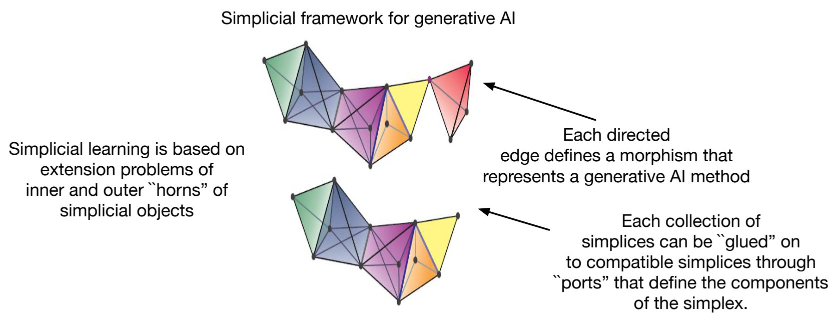

Simplicial framework for generative AI
Simplicial learning is based on extension problems of inner and outer "horns" of simplicial objects
Each directed edge defines a morphism that represents a generative AI method
Each collection of simplices can be "glued" on to compatible simplices through "ports" that define the components of the simplex.

Figure 13: DB and GT 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.

of the category `Param` defined in the previous chapter), 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}. 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 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 $[m] \rightarrow [n]$ 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 $\delta_i : [n] \rightarrow [n+1]$, called *coface* mappings, which omits $i \in [n]$, and a sequence of elementary surjections $\sigma_i : [n] \rightarrow [n-1]$, 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 $\Delta[n] \rightarrow X$. Every morphism $f : [n] \rightarrow [m]$ in Δ is functorially mapped to the map $\Delta[m] \rightarrow \Delta[n]$ 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 $\delta_i : [n-1] \rightarrow$

$[n], 0 \leq i \leq n$ is defined as:

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

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

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

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 $X(\delta_i) := d_i : X_n \to X_{n-1}$, and likewise, $X(\sigma_j) := s_j : X_{n-1} \to X_n$.

The compositions of these arrows define certain well-known properties:

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

Example 8. *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 9. *Given a category $\mathcal{C}$, we can identify an n-simplex σ of a simplicial set $\mathcal{C}_n$ with the sequence:*

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

the face operator d_0 applied to σ yields the sequence

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

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

Example 10. *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 11. *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 12. *Given a category $\mathcal{C}$, and an n-simplex σ of the simplicial set $\mathcal{C}_n$, the degeneracy operator $s_i, 0 \leq i \leq n$ 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 33. *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.*

Hierarchical Learning in DB and GT by solving Lifting Problems

DB and GT are based on a hierarchical model of simplicial learning, rather than the standard compositional learning framework. To give a deeper theoretical semantics of simplicial learning, 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.

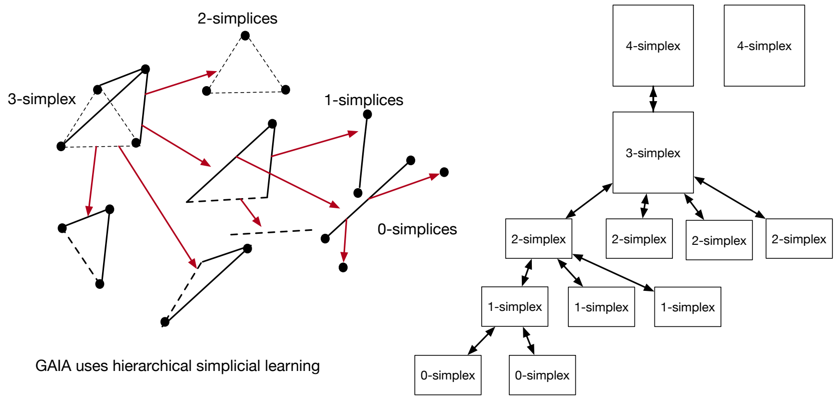


Figure 14: The hierarchical structure underlying DB and GT can be visualized as a "small business unit", defined as a 3-simplex that maintains its set of internal parameters, and updates them based on information it receives from its superiors and subordinates.

To understand how simplicial learning works, let us consider as an example the 3-simplex. 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 $\Delta^{op} \to \mathbf{Vect}$.

In DB, 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. 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 34. *Let $\mathcal{C}$ be a category. A* **lifting problem** *in $\mathcal{C}$ is a commutative diagram σ in $\mathcal{C}$.*

$$\begin{tikzcd} A \arrow[r, "\mu"] \arrow[d, "f"] & X \arrow[d, "p"] \\ B \arrow[r, "\nu"] & Y \end{tikzcd}$$

Definition 35. *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.*

$$\begin{tikzcd} A \arrow[r, "\mu"] \arrow[d, "f"] & X \arrow[d, "p"] \\ B \arrow[r, "\nu"] \arrow[ur, dashed, "h"] & Y \end{tikzcd}$$

Definition 36. *Let $\mathcal{C}$ be a category. If we are given two morphisms $f : A \to B$ and $p : X \to Y$ 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 $\mu : A \to X$ and $\nu : B \to Y$ satisfying the equations $p \circ \mu = \nu \circ f$, the associated lifting problem indicated in the diagram below.*

$$\begin{tikzcd} A \arrow[r, "\mu"] \arrow[d, "f"] & X \arrow[d, "p"] \\ B \arrow[r, "\nu"] \arrow[ur, dashed, "h"] & Y \end{tikzcd}$$

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

At its core, 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 13. *Given the paradigmatic non-surjective morphism $f : \emptyset \to \{\bullet\}$, any morphism p that has the right lifting property with respect to f is a* **surjective mapping**. .

$$\begin{tikzcd}
\emptyset \arrow[r, "\mu"] \arrow[d, "f"'] & X \arrow[d, "p"] \\
\{\bullet\} \arrow[r, "\nu"] \arrow[ur, dashed, "h"] & Y
\end{tikzcd}$$

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

Example 14. *Given the paradigmatic non-injective morphism $f : \{\bullet, \bullet\} \to \{\bullet\}$, any morphism p that has the right lifting property with respect to f is an* **injective mapping**. .

$$\begin{tikzcd}
\{\bullet, \bullet\} \arrow[r, "\mu"] \arrow[d, "f"'] & X \arrow[d, "p"] \\
\{\bullet\} \arrow[r, "\nu"] \arrow[ur, dashed, "h"] & Y
\end{tikzcd}$$

Simplicial Subsets and Horns in DB and GT

To explain how lifting problems form the computational substrate for DB and GT, we need to define lifting problems over n-simplicial complexes. The basic idea 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 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 37. *The* **standard simplex** Δ^n *is the simplicial set defined by the construction*

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

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

Definition 38. *Let $S_\bullet$ denote a simplicial set. If for every integer $n \geq 0$, 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 $\{T_n\}_{n \geq 0}$ defines a **simplicial subset** $T_\bullet \subseteq S_\bullet$

Definition 39. *The* **boundary** *is a simplicial set $(\partial\Delta^n) : \Delta^{op} \to$ **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 40. *The* **Horn** $\Lambda^n_i : \Delta^{op} \to$ **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 $0 \leq k \leq n$. 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.

$$\begin{tikzcd}
& \{0\} \arrow[dl] \arrow[dr] & \\
\{1\} \arrow[rr, dashed] & & \{2\}
\end{tikzcd}
\qquad
\begin{tikzcd}
& \{0\} \arrow[dl] \arrow[dr, dashed] & \\
\{1\} \arrow[rr] & & \{2\}
\end{tikzcd}$$

$$\begin{tikzcd}
& \{0\} \arrow[dl, dashed] \arrow[dr] & \\
\{1\} \arrow[rr] & & \{2\}
\end{tikzcd}$$

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.

Language Modeling With Geometric Transformers

In this chapter, we will give an empirical evaluation of DB and GT. To test whether the Geometric Transformer (GT) block can serve as a viable replacement for a standard Transformer block in a realistic language modeling setting, we conducted small-scale experiments on the WikiText-103 benchmark. The figure illustrates two variants of GT that we will test against a baseline Transformer in this section. For language modeling tasks, we instantiate this template on the 1D simplicial complex underlying a token sequence. Here:

- token positions $i = 1, \dots, L$ are the 0-simplices;
- edges $(i, i + 1)$ form the 1-skeleton (the local adjacency relation);
- the global operator is standard multi-head self-attention over $\{1, \dots, L\}$;
- the local geometric operator is implemented as a depthwise 1D convolution over the sequence, which performs learnable message-passing over the neighbors $\{i - 1, i, i + 1\}$; and
- the pointwise update is provided by a feed-forward network.

Experimental Results on WikiText-103

We now investigate the behavior of the Geometric Transformer (GT) block on a standard language modeling benchmark. Our goal is to empirically quantify how DB with GT compares to a baseline Transformer implementation in terms of optimization and parameter efficiency. We use WikiText-103 in the standard next-token prediction setting. The data are tokenized at the word level; the resulting vocabulary size is $|\mathcal{V}| \approx 2.7 \times 10^5$. We construct training samples by taking sliding windows of length $L = 128$ with stride 1. Models are trained with cross-entropy loss using AdamW, batch size 32, and a fixed learning rate schedule. All experiments use the same training loop and hyperparameters unless otherwise specified.

WIKITEXT-103 LANGUAGE MODELING. We evaluate the three architectures on the WikiText-103 benchmark using a comparable model size of $\sim 96M$ parameters. We keep the training setup intentionally simple: 4K context length, shared tokenization, and identical optimization hyperparameters across all models. We compare:

- **Transformer**: a standard decoder-only Transformer with L layers and model dimension d.
- **GeomTrans-Lite**: the same Transformer backbone augmented with a local geometric convolutional block (our "GT-Lite" layer) in each block.

- **GeomTrans-Full**: the full Geometric Transformer block, which replaces the standard MLP with our categorical / GT-based update.

We sweep over three depths, $L = 2, 4, 8$ with a fixed hidden size $d = 96$. All models are trained for 5,000 steps with the same optimizer, learning rate schedule, and batch size. The baseline Transformer converges slowly and plateaus with validation perplexity in the hundreds after 5,000 steps. In contrast, GeomTrans-Lite rapidly drives perplexity into the single-digit range, and GeomTrans-Full further reduces perplexity to values near 1.0. For $L = 4, 8$, the effect is even more pronounced. The baseline Transformer remains above 350 perplexity, while GeomTrans-Lite again reaches very low perplexity, and GeomTrans-Full consistently dominates both baselines throughout training, achieving the lowest validation perplexity at every checkpoint. In all cases, GeomTrans-Lite trains substantially faster than the baseline Transformer, and GeomTrans-Full yields the fastest convergence and lowest final loss. Importantly, the gap between GeomTrans-Lite and GeomTrans-Full grows with depth: at $L = 8$, the full GT block provides a clear additional gain over the lighter variant. For each configuration (dataset, depth, and width), we ran the Transformer, GeomTrans-Lite, and GeomTrans-Full models multiple times with different random seeds. We observed only minor variation in validation cross-entropy and perplexity across seeds, and in all cases the relative ranking between the three models (Transformer $>$ GeomTrans-Lite $>$ GeomTrans-Full in loss) remained unchanged. For visual clarity, the plots in the paper show a single representative run per configuration and omit error bars, since the spread was too small to materially affect our conclusions.

Constructing Large Causal Models

Domain	# Triangles
Economics and Finance	1140
Education and Human Capital	950
Legal and Forensic	553
Marketing and Product	1059
Medicine and Healthcare	932
Operations and Engineering	1281
Public Health and Environment	1336
Social Sciences and Policy	1015
Technology and AI	782

Table 2: Simplicial complex constructed from a $100K$ causal claims dataset.

We briefly describe an application of using DB and GT to build massively large causal models (LCMs) from carefully curated queries to a large language model (LLM). We define a structured ontology containing ten coarse domains (*Economics and Finance*, *Medicine and Healthcare*, *Marketing and*

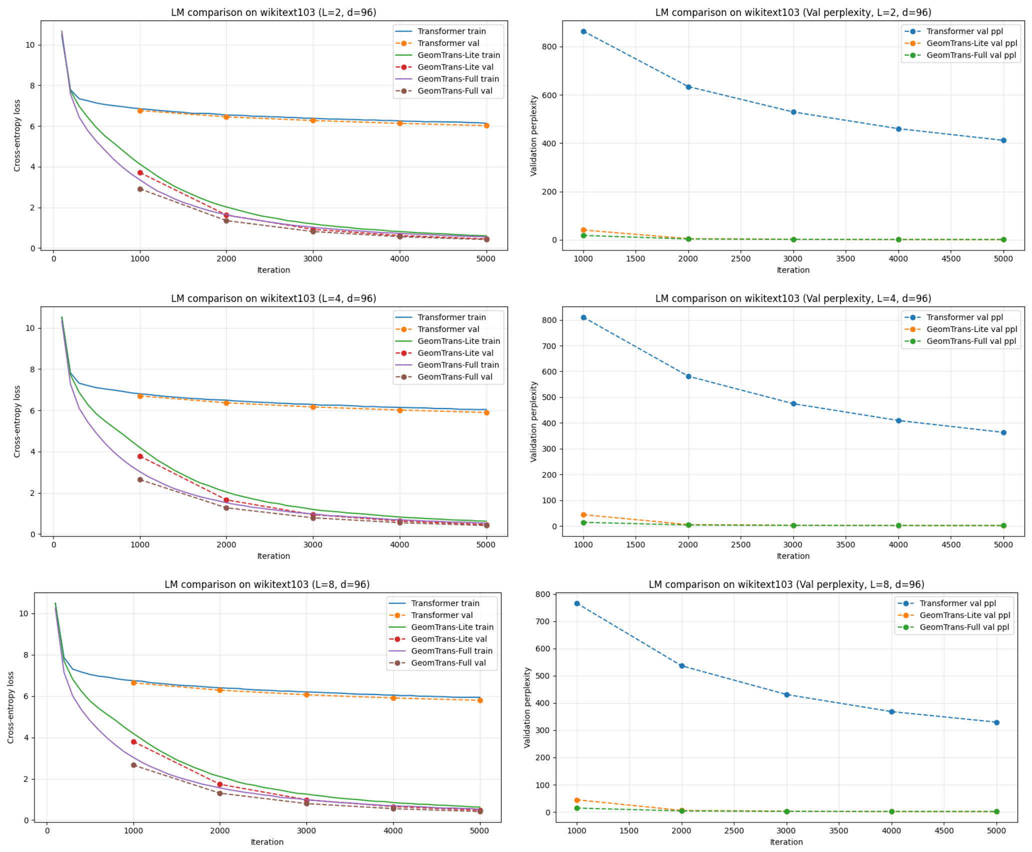


Figure 15: **WikiText-103 language modeling**: Validation perplexity (right column) and train/validation cross-entropy loss (left column) for Transformer, GeomTrans-Lite, and GeomTrans-Full at depths $L = 2$ (top), $L = 4$ (middle), and $L = 8$ (bottom). In all settings, GeomTrans-Lite dramatically outperforms the baseline Transformer, and GeomTrans-Full consistently attains the lowest loss and perplexity. All models are of size $\sim 96M$ parameters and trained with identical parameters, and their running times are roughly comparable.

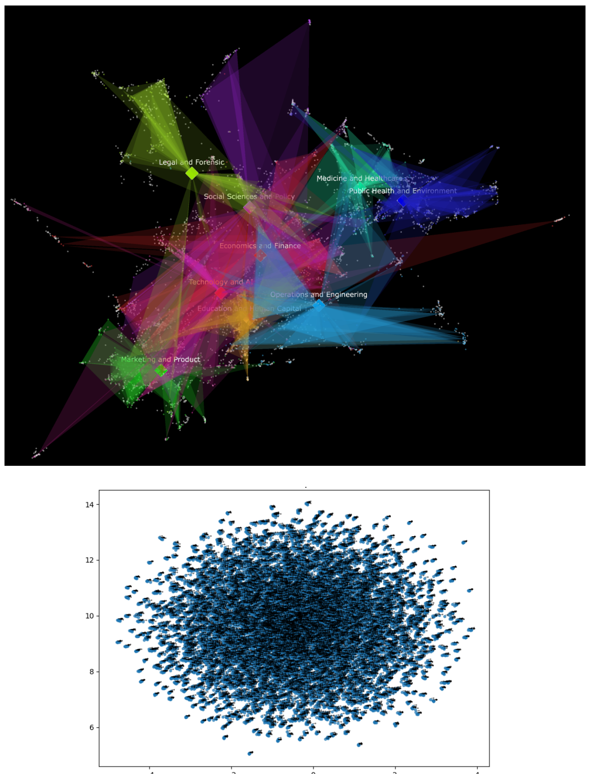


Figure 16: Top: $3D$ UMAP visualization of the $\sim$ $100K$ causal claims manifold after processing by DB and GT. Bottom: UMAP manifold visualization on the normalized vectors in $\mathbb{R}^{384}$ produced by a baseline `all-MiniLM-L6-v2` SentenceTransformer with no post-processing by DB or GT.

Product, etc.) and one hundred fine-grained subtopics (ten per domain). We use $N \in \{50, 100, 500, 1000\}$ queries for each domain, depending on the scale of the experiment. For the full-scale relational corpus we use $N = 1000$ for each of 100 subtopics, yielding 100,000 sentences. After deduplication, our final dataset contained 90,016 statements. Each extracted triple is associated with the coarse domain from the corpus metadata. Across the 90K dataset we obtain 54,514 unique entities and 57,390 relational edges. We embed each unique entity using the `all-MiniLM-L6-v2` SentenceTransformer, yielding normalized vectors in $\mathbb{R}^{384}$. No fine-tuning is performed. Cosine similarity is used throughout. For each domain, we form 2-simplices (triangles) by connecting all triples (h, r, t_i) and (h, r, t_j) that share the same head entity h. This produces domain-local simplicial patches reflecting the latent relational neighborhood.

Geometric Refinement via DB and GT

We apply three iterations of diagrammatic message passing using the Geometric Transformer (GT). GT in this specific experiment was used *without* supervised training: all weights are random but fixed, and no gradients are computed. This yields a smoothed embedding matrix $\mathbf{V}_{\text{refined}}$. The results of GT and DB processing are shown in the above figure. As a sharp contrast to this rich manifold structure produced by GT and DB processing, the figure shows the UMAP embedding on the same causal relational triples produced by a baseline `all-MiniLM-L6-v2` SentenceTransformer revealing no structure at all, but an isotropic Gaussian-like blob of points.

A Global–Local Operator Perspective

A standard Transformer layer implements a *global* operator on this complex: self-attention defines a data-dependent kernel $K_{\text{attn}}(i, j; x)$ over all positions $i, j \in \{1, \ldots, L\}$, and applies a global mixing step of the form

$$h_i^{\text{attn}} \approx \sum_{j=1}^{L} K_{\text{attn}}(i, j; x)\, h_j,$$

followed by a pointwise nonlinearity in the FFN. In contrast, our depth-wise Conv1d layer realizes a purely *local* geometric operator K_{geo} on the 1-skeleton of X, mixing only a fixed neighborhood $\mathcal{N}(i)$ (e.g. $\{i-1, i, i+1\}$). At a high level,

$$h_i^{\text{geo}} \approx \sum_{j \in \mathcal{N}(i)} K_{\text{geo}}(i, j)\, h_j,$$

where K_{geo} can be viewed as a learnable discretization of a Laplacian or diffusion operator on the sequence graph. GT layers combine both of these operators in a single block.

Summary and Further Reading

In this chapter, we introduced a novel Geometric Transformer (GT) model that works by "gluing" together local solutions in a way that is arguably superior to a traditional Transformer. A deeper study of how GT works is given in the next chapter, where we introduce the concept of *dynamic compositionality*. There are many deeper connections to algebraic topology and cohomology theory that we do not have space to discuss, and are outside the scope of this introductory book.

It is highly recommended that the reader take time to understand the basic ideas in topological data analysis, including notions like *persistence homology*. [52]

[52] Herbert Edelsbrunner, David Letscher, and Afra Zomorodian. Topological persistence and simplification. *Discrete & Computational Geometry*, 28(4):511–533, 2002

Dynamic Compositionality

The Transformer architecture is explicitly compositional at the level of the forward graph. However, we show that its learning dynamics under gradient descent are generally not compositional, even for simple residual blocks. This paper explains the mechanism behind this discrepancy and connects it directly to the instrumentation used in Diagrammatic Backpropagation (DB) and Geometric Transformer (GT) described in the previous two chapters. We analyze the commutator-energy probe—computed as the mean-squared difference between $T_1(T_2(x))$ and $T_2(T_1(x))$ for sub-operators T_1, T_2 —and show how it quantifies order-sensitivity of learned representation updates on the states visited during training. We then explain how geometric inductive biases in Geometric Transformers (local smoothing, geometric transport, and mixture routing) systematically reduce this order-sensitivity by improving compatibility between sub-operator deformations, thereby stabilizing learning. Our goal is to provide a code-faithful, mechanistic account of DB+GT that complements prior empirical results and serves as documentation for open-source releases.

Introduction

Modern neural architectures are built compositionally: complex models are assembled by stacking reusable blocks such as attention, feed-forward maps, normalization layers, and residual connections. This compositional structure is explicit in the forward computation graph of the Transformer. Yet, empirical evidence suggests that the learning dynamics induced by gradient descent do not necessarily respect this compositional structure. In practice, the behavior of a trained model can become highly sensitive to interactions between sub-operators within a block, leading to instability and poor generalization.

Diagrammatic Backpropagation (DB) along with a family of Geometric Transformers (GTs) introduced in the previous two chapters systematically reduce a proxy diagnostic we call commutator energy, while a baseline Transformer often does not. Beyond task-level metrics, DB and GT also introduced a set of instrumental probes that combine algebraic diagnostics inspired by Čech cohomology. [53]. We also use complementary tools from topological

[53] Raoul Bott and Loring W. Tu. *Differential Forms in Algebraic Topology*. Springer, 1982; and Robert Ghrist. *Elementary Applied Topology*. Createspace, 2014

data analysis [54] enabling direct measurement of internal compositional compatibility and global attention geometry.

In this paper, we provide a detailed explanation of DB and GT by focusing on their mechanics “under the hood.” Our focus is not on new benchmarks, but on explaining why the following code-level measurement is meaningful and why GT architectures drive it down:

$$\mathrm{Comm}(T_1, T_2; x) \; = \; \|T_1(T_2(x)) - T_2(T_1(x))\|_2^2,$$

where T_1, T_2 are concrete sub-operators (e.g., attention, feed-forward, convolutional smoothing, geometric transport, or MoE mixing) evaluated on representations x encountered during training. We argue that commutator energy quantifies order-sensitivity of learned representation updates and serves as a practical proxy for gradient interference and lack of dynamic compositionality.

[54] Herbert Edelsbrunner, David Letscher, and Afra Zomorodian. Topological persistence and simplification. *Discrete & Computational Geometry*, 28(4):511–533, 2002

Definition 41. *(***Dynamic compositionality (informal)***): A model is statically compositional if its forward computation is a composition of modules. It is dynamically compositional if the learning-induced deformations of these modules are mutually compatible on the representations visited during training, so that training is not overly sensitive to the effective order in which sub-operator updates interact.*

COMMUTATOR ENERGY measures dynamic compositionality failure by quantifying the discrepancy between two compositions applied to the same representation x. We do not require modules to commute as functions; rather, we use commutator energy as an empirical diagnostic of order sensitivity and operator incompatibility under learning.

Čech-style obstruction proxy

Performance curves alone do not explain *why* the baseline Transformer plateaus while GT continues improving. We therefore introduce a computable proxy for the local-to-global inconsistency obstruction. We treat each encoder block as a collection of local “patch maps” (e.g., attention A, feed-forward F, and in GT variants an additional geometric operator C or G) acting on the shared residual stream x. For a patch map $T_i(x) = x + \Delta_i(x)$, global coherence implies approximate commutativity of local updates around loops. We quantify loop inconsistency using the normalized commutator energy

$$\mathrm{Comm}(i, j; x) = \frac{\|T_i(T_j(x)) - T_j(T_i(x))\|^2}{\|x\|^2 + \varepsilon}.$$

The *Čech obstruction proxy* for a layer is the average commutator energy over relevant patch pairs; the model-level proxy is the average over encoder layers. Concretely, we use:

- Transformer: average of $\mathrm{Comm}(A, F; x)$,
- GT-Lite: average of $\mathrm{Comm}(A, C; x), \mathrm{Comm}(C, F; x), \mathrm{Comm}(A, F; x)$,
- GT-Full: average of $\mathrm{Comm}(A, F; x), \mathrm{Comm}(F, G; x), \mathrm{Comm}(A, G; x)$,

computed in evaluation mode to remove stochastic effects from dropout. This obstruction proxy is a metricized Čech-1 inconsistency: it measures whether locally defined updates can be glued into a globally coherent transformation on the overlap interface.

A Minimal Demonstration: Residual MLPs on Two Moons

To illustrate that commutator energy is not specific to Transformers or attention-based models, we construct a minimal demonstration using a simple residual multilayer perceptron (MLP) trained on the classic two-moons classification task. This example serves both as a pedagogical illustration and as a smoke test for Diagrammatic Backpropagation (DB) and Geometric Transformer (GT) principles in their simplest form.

SETUP. We consider three architectures with identical capacity and training hyperparameters: (i) a baseline residual MLP composed of stacked residual blocks, (ii) the same MLP augmented with a GT-Lite alignment module that performs local feature-space smoothing, and (iii) the same MLP augmented with a GT-Full alignment module that performs explicit transport via message passing on a fixed feature graph. All models are trained to perfect accuracy on the two-moons task within a few epochs.

COMMUTATOR-ENERGY PROBE. During training, we measure the normalized commutator energy between adjacent modules in the residual stack:

$$\mathrm{Comm}(T_i, T_{i+1}; x) = \frac{\|T_i(T_{i+1}(x)) - T_{i+1}(T_i(x))\|_2^2}{\|x\|_2^2 + \varepsilon},$$

averaged over module pairs and minibatches. This quantity captures order sensitivity between learned sub-operators and is computed purely in the forward pass, independent of task loss.

RESULTS. The figure below shows the evolution of commutator energy over training. Although all three models achieve identical task performance, their internal dynamics differ markedly. The baseline residual MLP exhibits steadily increasing commutator energy, indicating growing order sensitivity despite convergence of the task loss. In contrast, the GT-Lite variant maintains commutator energy an order of magnitude lower, with only mild drift. The GT-Full variant achieves the lowest and most stable commutator energy throughout training.

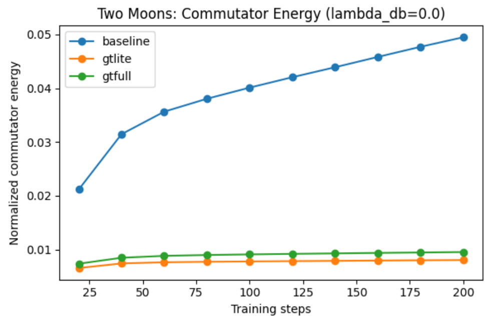


Figure 17: Commutator energy during training on the two-moons task. All models reach perfect accuracy, but the baseline residual MLP exhibits steadily increasing order sensitivity. GT-Lite reduces commutator energy by local smoothing, while GT-Full achieves the lowest and most stable values via explicit transport.

INTERPRETATION. This simple experiment demonstrates that dynamic compositionality failure is a general phenomenon of gradient-based learning in compositional architectures, not an artifact of attention or sequence modeling. Local smoothing (GT-Lite) partially mitigates order sensitivity, while explicit transport and alignment (GT-Full) suppress it almost entirely. Importantly, these differences are invisible to standard task metrics, underscoring the value of commutator energy as an instrumental probe of learning-time behavior.

Results: obstruction trajectories explain stability

The next figure plots the obstruction proxy during training for the three models on WikiText-103 dataset. The results reveal three qualitatively different regimes. The baseline Transformer accumulates increasing obstruction over training, indicating growing loop inconsistency among local updates; this coincides with degraded validation performance. GeomTrans-Lite initially reduces obstruction, consistent with improved early generalization, but later exhibits *obstruction drift* in which commutator energy rises as training proceeds, correlating with reduced validation gains. In contrast, GeomTrans-Full maintains low and remarkably stable obstruction throughout training, matching its sustained improvements in validation perplexity. These findings support the interpretation of GT as a descent architecture: GT-Full enforces higher-order overlap consistency strongly enough to prevent inconsistency accumulation, whereas GT-Lite enforces only local overlap constraints and can drift at longer compositional scales. Our goal in this paper is to explain the underlying machinery that makes GT models suprisingly good at enforcing dynamic compositionality.

Dynamic Compositionality and Order Sensitivity in Residual Learning

Deep networks are *statically compositional*: their forward computation is a composition of modules arranged in a directed acyclic graph. For example,

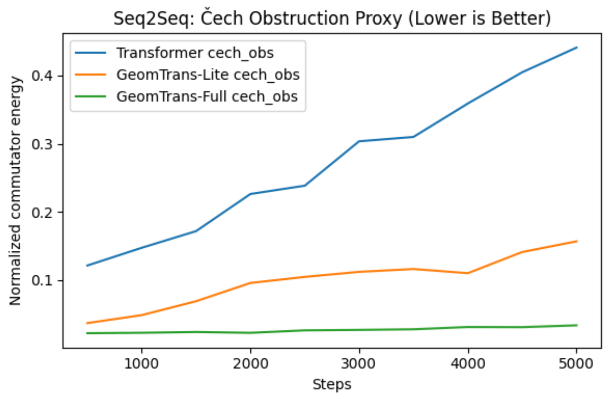


Figure 18: **Čech obstruction proxy during training on WikiText-103.** Normalized commutator energy (lower is better) for baseline Transformer, GT-Lite, and GT-Full. The baseline Transformer exhibits steadily increasing obstruction (inconsistency accumulation). GT-Lite reduces obstruction early but later drifts upward. GT-Full maintains low, stable obstruction throughout training, aligning with its superior validation performance.

a Transformer block composes attention, residual additions, normalization, and a feed-forward subnetwork. Static compositionality is guaranteed by architecture. In contrast, this paper is concerned with *dynamic compositionality*: whether the *learning dynamics* induced by gradient descent respect the intended modular decomposition. Our central observation is that, even when the forward graph is perfectly compositional, the learned transformations produced by different sub-operators can interact in a strongly order-sensitive way on the representations encountered during training. This order sensitivity is a practical signature of "non-compositional" learning dynamics and is the quantity that Diagrammatic Backpropagation (DB) seeks to measure and control.

From static composition to learned deformations

Residual architectures provide a natural entry point because each sublayer is explicitly an identity map plus a learned correction. Consider two residual modules T_1 and T_2 acting on a representation $x \in \mathbb{R}^d$:

$$T_1(x) = x + f_1(x), \qquad T_2(x) = x + f_2(x),$$

where f_1 and f_2 are learnable maps (e.g., an attention transform, a feed-forward network, a convolutional smoother, or a geometric transport map). This form does *not* assume that modules should commute as functions. It only highlights that learning introduces state-dependent *deformations* of the representation space through f_1 and f_2.

Although the forward computation is unambiguous (the architecture fixes a specific order), learning can be sensitive to the interaction between these deformations: applying T_1 changes the input seen by T_2, and vice versa. When these interactions are mild, the model behaves as if its modules are well-coordinated. When interactions are strong, the learning dynamics become fragile: small changes in one submodule can dramatically alter the effect of another.

Order sensitivity and commutator energy

To make this notion of interaction operational, we introduce a forward-only probe that measures *order sensitivity* on representations encountered during training. Given two sub-operators T_1 and T_2, define the *commutator residual* at x as

$$r(T_1, T_2; x) \;=\; T_1(T_2(x)) - T_2(T_1(x)).$$

We then define the corresponding *commutator energy* as the mean-squared magnitude of this residual, optionally normalized by the input energy:

$$E_{\text{comm}}(T_1, T_2; x) \;=\; \frac{\|r(T_1, T_2; x)\|_2^2}{\|x\|_2^2 + \varepsilon}.$$

In our implementation, this is computed exactly as:

$$E_{\text{comm}}(T_1, T_2; x) \;\propto\; \text{MSE}\big(T_1(T_2(x)) - T_2(T_1(x))\big),$$

matching the code fragment `mse(T1(T2(x)) - T2(T1(x)))`. Importantly, we do *not* interpret small commutator energy as a requirement that T_1 and T_2 commute as abstract functions. Rather, commutator energy is used as an empirical proxy for how strongly two learned deformations interfere on the states the model actually visits. Large values indicate that the effect of one sub-operator substantially changes the behavior of the other, implying a lack of *exchangeability* in their learned updates.

WHY ORDER SENSITIVITY MATTERS FOR LEARNING. Even though E_{comm} is computed using only forward evaluations, it tracks a key failure mode of deep learning systems: *gradient interference* between interacting modules. When the representation space is such that sub-operators induce highly order-sensitive transformations, optimization signals propagated through different subpaths can conflict, producing unstable training and poor generalization. Conversely, when sub-operators are more compatible on the data manifold, training becomes more stable and the model can support coherent global structure. In this sense, dynamic compositionality is not about enforcing commutativity, but about controlling destructive order-sensitivity in learned module interactions.

A minimal residual example

To build intuition, consider two residual maps $T_1(x) \;=\; x + f_1(x)$ and $T_2(x) \;=\; x + f_2(x)$ applied to the same input x. The two compositions expand as

$$\begin{aligned} T_1(T_2(x)) &= x + f_2(x) + f_1(x + f_2(x)), \\ T_2(T_1(x)) &= x + f_1(x) + f_2(x + f_1(x)). \end{aligned}$$

Their difference arises entirely from the *state dependence* of the learned deformations: $f_1(\cdot)$ and $f_2(\cdot)$ are evaluated at different perturbed inputs.

When these perturbations strongly change the behavior of the other module, commutator energy is large. When their effects are compatible, commutator energy is small. This provides an intuitive explanation for why commutator energy is a useful diagnostic even for architectures whose forward graph is fixed: it measures whether the learned sub-operators act as well-coordinated "editors" of the representation space.

ROADMAP. We instantiate these definitions inside Transformer encoder blocks by taking T_1, T_2 to be concrete sub-stages such as attention, feed-forward, convolutional smoothing, geometric transport, and mixture-of-experts mixing. We show that Geometric Transformer variants systematically reduce commutator energy relative to a baseline Transformer, providing a mechanistic account of the instrumental probes reported in the v1 paper.

A Minimal Residual MLP Example: Order Sensitivity in Learning Dynamics

To isolate the mechanism underlying commutator energy without architectural complexity, we consider a minimal residual multilayer perceptron (MLP) example. The goal of this section is not to achieve high performance, but to demonstrate that order sensitivity arises naturally in learning dynamics even in the simplest residual setting. This example serves as a sanity check and a conceptual bridge between abstract definitions and Transformer-scale models.

Two interacting residual blocks

Let $x \in \mathbb{R}^d$ be an input representation and consider two residual modules applied sequentially:

$$T_1(x) = x + f_1(x), \qquad T_2(x) = x + f_2(x),$$

where f_1 and f_2 are small MLPs with independent parameters. The composite forward computation is fixed by architecture, for example $T_2(T_1(x))$. Statically, this model is compositional: it is simply a composition of two functions.

During training, however, f_1 and f_2 are updated simultaneously by gradient descent on a shared loss. Each module therefore induces a learned, state-dependent deformation of the representation space. The interaction between these deformations determines whether learning is dynamically compositional.

Order sensitivity in residual updates

Although the architecture applies T_1 before T_2, we can probe the learning dynamics by comparing the two compositions

$$T_2(T_1(x)) \quad \text{and} \quad T_1(T_2(x)).$$

Expanding these expressions reveals the source of order sensitivity:

$$\begin{aligned} T_2(T_1(x)) &= x + f_1(x) + f_2(x + f_1(x)), \\ T_1(T_2(x)) &= x + f_2(x) + f_1(x + f_2(x)). \end{aligned}$$

The difference between these two outputs arises entirely from the *state dependence* of f_1 and f_2. When the perturbation induced by one module significantly alters the input seen by the other, the two compositions differ substantially. This discrepancy is precisely what the commutator-energy probe measures:

$$E_{\text{comm}}(T_1, T_2; x) = \|T_2(T_1(x)) - T_1(T_2(x))\|_2^2.$$

Importantly, large commutator energy does not indicate a violation of any architectural constraint. Rather, it indicates that the two learned deformations interact in an order-sensitive way on the states encountered during training.

Connection to gradient interference

The order sensitivity measured by commutator energy has a direct interpretation in terms of learning dynamics. When f_1 and f_2 induce incompatible deformations, small changes in the parameters of one module can dramatically alter the effective gradient signal seen by the other. This manifests as *gradient interference*: updates that improve the loss along one path can degrade it along another.

From this perspective, commutator energy serves as a forward-only proxy for detecting gradient interference. High commutator energy indicates that the learning-induced updates of different modules are poorly coordinated, while low commutator energy indicates approximate exchangeability of updates and more stable learning dynamics.

Implications for Diagrammatic Backpropagation

Diagrammatic Backpropagation can be interpreted as introducing an auxiliary control objective that discourages large order sensitivity in learned deformations. In the residual MLP setting, minimizing commutator energy promotes compatibility between f_1 and f_2, without requiring them to commute or collapse to trivial functions. This stabilizes learning by reducing sensitivity to update order while preserving expressivity.

Although this example is deliberately minimal, it captures the essential mechanism that appears in larger architectures. Transformer blocks replace

f_1 and f_2 with attention, feed-forward, and geometric transport operators, but the source of order sensitivity—and the role of commutator energy in controlling it—remains the same. The residual MLP thus provides a transparent microcosm of the dynamic compositionality issues addressed by Diagrammatic Backpropagation.

Dynamic Compositionality in Transformer and Geometric Transformer Blocks

Previously we introduced the notion of dynamic compositionality and argued that order sensitivity of learned sub-operators is a practical signature of non-compositional learning dynamics. We now instantiate this analysis concretely inside Transformer encoder blocks and their geometric variants, showing how the commutator-energy probe used in the v1 paper arises directly from interactions between familiar architectural components.

Transformer encoder blocks as interacting sub-operators

A standard Transformer encoder block consists of a small number of sub-stages applied sequentially to the same representation: multi-head self-attention, residual addition, layer normalization, and a positionwise feed-forward network. Although the execution order is fixed by the architecture, learning dynamics depend on how these sub-stages interact on the representation manifold visited during training.

For the purposes of dynamic compositionality analysis, we group each sub-stage together with its residual connection and normalization into a single effective sub-operator. In the baseline Transformer, we therefore identify two concrete sub-operators:

- A: the attention stage, including residual addition and layer normalization;
- F: the feed-forward stage, including residual addition and layer normalization.

These are exactly the operators implemented in the probe as: `A(z) = LN(z + Attn(z))` and `F(z) = LN(z + FF(z))`.

Measuring order sensitivity inside a block

Although the Transformer block applies A before F in the forward pass, the learning dynamics depend on how strongly these sub-operators perturb each other's effective inputs. To probe this interaction, we evaluate the commutator-energy diagnostic

$$E_{\text{comm}}(A, F; x) = \frac{\|A(F(x)) - F(A(x))\|_2^2}{\|x\|_2^2 + \varepsilon},$$

computed on the actual block input x encountered during training. This quantity measures how sensitive the learned representation update is to the order in which attention and feed-forward transformations are applied. Importantly, we do not require A and F to commute as functions. Rather, we interpret large commutator energy as evidence that the two sub-operators induce incompatible state-dependent deformations, making learning dynamics sensitive to update order.

Why vanilla Transformers exhibit high commutator energy

In a baseline Transformer, self-attention introduces global, token-coupling effects that depend sharply on the current representation, while the feed-forward network applies nonlinear, positionwise transformations. Layer normalization and residual connections tightly couple these effects across feature dimensions. As a result, applying attention significantly alters the input distribution seen by the feed-forward network, and vice versa. This produces strong order sensitivity between A and F on the states visited during training, leading to high commutator energy.

Empirically, this manifests as increasing commutator energy over training and correlates with instability and poor generalization at scale. The probe thus captures a concrete failure mode of dynamic compositionality in vanilla Transformers: sub-operators act as poorly coordinated "editors" of the same representation.

Geometric Transformers

We introduced a family of GT models in the previous chapter, which we now analyze from the standpoint of dynamic compositionality.

GT-Lite: reducing order sensitivity via local smoothing

GT-Lite augments the Transformer block with an additional sub-operator C that performs local geometric smoothing, implemented as a convolution or local neighborhood mixer. In this case, the probe measures the average of pairwise order sensitivities:

$$E_{\text{comm}}(A, C; x), \quad E_{\text{comm}}(C, F; x), \quad E_{\text{comm}}(A, F; x).$$

Local smoothing reduces high-frequency variation in representations and regularizes the representation manifold seen by downstream sub-operators. As a result, the effects of attention and feed-forward transformations become more compatible, and order sensitivity is reduced. This explains why GT-Lite consistently exhibits lower commutator energy than a baseline Transformer, especially in early and mid stages of training.

GT-Full: geometric transport and alignment of representation geometry

The full Geometric Transformer replaces local smoothing with an explicit geometric transport operator G, implemented via simplicial lifting, message passing, and readout over a token-interaction complex. The corresponding probe measures:

$$E_{\text{comm}}(A, G; x), \quad E_{\text{comm}}(F, G; x), \quad E_{\text{comm}}(A, F; x).$$

Geometric transport aligns local coordinate frames across tokens by propagating information along edges and higher-order simplices. This alignment reduces curvature mismatch between the deformations induced by attention and feed-forward sub-operators. Consequently, GT-Full exhibits the lowest and most stable commutator energy across tasks and scales, while still supporting rich global structure in representations. Mechanistically, GT-Full achieves dynamic compositionality not by suppressing expressivity, but by ensuring compatibility between learned deformations.

GT-MoE: routing-induced order sensitivity

In GT-MoE, the feed-forward stage is replaced by a mixture-of-experts operator M whose effective transformation depends on gating decisions computed from the current representation. The probe measures $E_{\text{comm}}(A, M; x)$. Because routing introduces additional state-dependent branching, the interaction between attention and mixture selection is inherently more order-sensitive. This explains why GT-MoE typically exhibits higher or more variable commutator energy than GT-Full, while still improving over a baseline Transformer.

Summary: architectural control of dynamic compositionality

Across all variants, commutator energy serves as a unified diagnostic for dynamic compositionality inside Transformer blocks. Vanilla Transformers exhibit high order sensitivity due to incompatible sub-operator interactions. GT-Lite reduces this sensitivity through local smoothing, while GT-Full achieves the strongest reduction by aligning representation geometry via explicit transport. These mechanistic differences directly explain the empirical commutator-energy trends reported in the previous chapter and motivate Diagrammatic Backpropagation as a control mechanism for learning-time compositionality.

From Mechanism to Implementation: Explaining the DB+GT Code Path

Sections – established that commutator energy provides a forward-only diagnostic of dynamic compositionality failure, measuring order sensitivity between learned sub-operators. We now complete the analysis by explaining how this diagnostic is implemented in practice for Transformer and Geometric Transformer models, and why the observed empirical behavior follows directly from the code structure. Our goal in this section is to demystify the DB+GT implementation by mapping each conceptual component to the corresponding computational step.

The commutator-energy primitive

At the lowest level, the DB instrumentation relies on a single primitive:

$$\mathrm{Comm}(T_1, T_2; x) \; = \; \|T_1(T_2(x)) - T_2(T_1(x))\|_2^2,$$

implemented directly as:

```
_mse(T1(T2(x)) - T2(T1(x)))
```

This computation involves no gradients and introduces no additional parameters. It is evaluated on the same representations encountered during training and serves purely as a measurement of order sensitivity. Normalizing by $\|x\|_2^2 + \varepsilon$ ensures that the resulting value is scale-free and comparable across layers, models, and training stages.

Crucially, this primitive is agnostic to the nature of T_1 and T_2; they are concrete callable subroutines corresponding to actual sub-stages of a model (e.g., attention, feed-forward, convolutional smoothing, geometric transport, or mixture-of-experts routing).

Layerwise instrumentation in Transformer blocks

The function `encoder_cech_obstruction` evaluates commutator energy inside each encoder block by extracting its internal sub-operators. For a baseline Transformer block, these are:

- A: the self-attention stage together with residual addition and layer normalization;
- F: the feed-forward stage together with residual addition and layer normalization.

The instrumentation computes

$$E_{\mathrm{comm}}(A, F; x_{\mathrm{in}})$$

on the block input x_{in} and then advances the forward pass normally. The resulting scalar measures how strongly attention and feed-forward updates interfere on the representations encountered at that layer.

Because this computation is performed under `@torch.no_grad()`, it does not affect training unless explicitly included as a loss term. Instead, it provides a per-layer diagnostic of dynamic compositionality.

Extending the probe to Geometric Transformer variants

Geometric Transformer variants introduce additional sub-operators, and the instrumentation generalizes by measuring pairwise order sensitivity among them.

GT-LITE. GT-Lite augments the block with a local geometric smoothing operator C (e.g., a 1D convolution over neighboring tokens). The probe evaluates the average of:

$$E_{\text{comm}}(A, C; x), \quad E_{\text{comm}}(C, F; x), \quad E_{\text{comm}}(A, F; x),$$

capturing overall compatibility among attention, smoothing, and feed-forward updates. Local smoothing regularizes the representation manifold, reducing sharp state dependence and lowering order sensitivity.

GT-FULL. The full Geometric Transformer introduces an explicit geometric transport operator G implemented via simplicial lifting, message passing, and readout. The probe evaluates:

$$E_{\text{comm}}(A, G; x), \quad E_{\text{comm}}(F, G; x), \quad E_{\text{comm}}(A, F; x).$$

Geometric transport aligns local coordinate frames across tokens, substantially reducing curvature mismatch between learned deformations. As a result, GT-Full exhibits consistently low commutator energy while retaining expressive global structure.

GT-MOE. In GT-MoE, the feed-forward stage is replaced by a mixture-of-experts operator M whose effective transformation depends on routing probabilities. The probe measures $E_{\text{comm}}(A, M; x)$, capturing order sensitivity introduced by state-dependent expert selection. This explains the intermediate and sometimes variable commutator-energy behavior observed empirically.

Why commutator energy tracks training stability

Although commutator energy is computed using forward evaluations, it correlates strongly with training stability because it reflects the compatibility of learning-induced deformations. When commutator energy is large, small changes in one sub-operator substantially alter the effective behavior of others, amplifying gradient interference and making optimization sensitive to update order. When commutator energy is small, sub-operators act as approximately exchangeable contributors to representation updates, yielding more stable and predictable learning dynamics.

This explains why baseline Transformers often exhibit rising commutator energy during training, while Geometric Transformer variants systematically drive it down. DB does not require exact commutativity or impose algebraic constraints on the architecture; it instead penalizes destructive order sensitivity, allowing models to retain expressivity while improving coordination.

From instrumentation to control

In the previous chapter, commutator energy is primarily used as an instrumental probe. When Diagrammatic Backpropagation is enabled, the same quantity (or its generalization to higher-order overlap patterns) is incorporated into the training objective as an auxiliary loss. This transforms a passive diagnostic into an active control mechanism that directly shapes learning dynamics.

From an implementation standpoint, DB training differs from standard backpropagation only by the addition of these overlap-consistency losses and their gradients. All core architectural components remain unchanged. The resulting behavior—reduced commutator energy, improved stability, and the emergence of coherent global structure—follows directly from minimizing order sensitivity of learned sub-operator interactions.

SUMMARY. This section has shown that the DB+GT implementation is neither mysterious nor ad hoc. It operationalizes a simple idea: learning should not depend strongly on the order in which compatible sub-operators are applied. The commutator-energy probe provides a concrete, code-level measurement of this principle, and Geometric Transformer architectures supply the geometric scaffolding needed to enforce it effectively.

GT-Lite Under the Hood: Transformer Blocks with Local Geometric Smoothing

We now unpack the simplest Geometric Transformer variant, *GT-Lite*, at the level of concrete PyTorch code. The purpose of this section is to explain how a small architectural modification—a local geometric smoothing operator inserted between attention and feed-forward stages—systematically reduces commutator energy and improves dynamic compositionality. This section is intended to serve as a code-level reference for readers wishing to understand or extend the DB+GT implementation.

GeomEncoderBlock: structure and intent

Listing 1 shows the implementation of the GT-Lite encoder block. Compared to a standard Transformer encoder block, the only substantive addition is a one-dimensional convolution applied along the token dimension, which performs local geometric smoothing.

Listing 1: GT-Lite encoder block

```
class GeomEncoderBlock(nn.Module):
    def __init__(self, d_model: int, n_heads: int, dim_ff: int, dropout: float = 0.1):
        super().__init__()
        self.self_attn = nn.MultiheadAttention(
            d_model, n_heads, dropout=dropout, batch_first=True
        )
        self.conv = nn.Conv1d(d_model, d_model, kernel_size=3, padding=1)

        self.lin1 = nn.Linear(d_model, dim_ff)
        self.lin2 = nn.Linear(dim_ff, d_model)

        self.ln_attn = nn.LayerNorm(d_model)
        self.ln_conv = nn.LayerNorm(d_model)
        self.ln_ff   = nn.LayerNorm(d_model)

        self.dropout = nn.Dropout(dropout)
        self.dropout_ff = nn.Dropout(dropout)

    def forward(self, x: torch.Tensor) -> torch.Tensor:
        attn_out, _ = self.self_attn(x, x, x, need_weights=False)
        x = self.ln_attn(x + self.dropout(attn_out))

        x_conv = self.conv(x.transpose(1, 2)).transpose(1, 2)
        x = self.ln_conv(x + 0.2 * x_conv)

        ff = self.lin2(self.dropout_ff(F.relu(self.lin1(x))))
        x = self.ln_ff(x + ff)
        return x
```

Sub-operators and dynamic compositionality

From the perspective of dynamic compositionality, the GT-Lite encoder block consists of three interacting sub-operators, each applied with a residual connection and layer normalization:

- **Attention operator** A: global token mixing via multi-head self-attention,
$$A(x) = \mathrm{LN}_{attn}(x + \mathrm{Attn}(x)).$$
- **Local smoothing operator** C: convolution over neighboring token positions,
$$C(x) = \mathrm{LN}_{conv}(x + \alpha\,\mathrm{Conv}(x)), \quad \alpha = 0.2.$$
- **Feed-forward operator** F: positionwise nonlinear deformation,
$$F(x) = \mathrm{LN}_{ff}(x + \mathrm{FF}(x)).$$

The overall block computes $F(C(A(x)))$. Architecturally, this is a fixed composition. Dynamically, however, learning depends on how strongly these sub-operators interfere when applied to the same representation.

Why local smoothing reduces commutator energy

The commutator-energy probe evaluates order sensitivity between sub-operators, for example

$$E_{\mathrm{comm}}(A, F; x) = \|A(F(x)) - F(A(x))\|_2^2.$$

In a baseline Transformer, A and F interact directly. Attention introduces sharp, token-dependent changes to the representation, and the feed-forward network is highly sensitive to such changes. As a result, $A(F(x))$ and $F(A(x))$ often differ substantially, producing large commutator energy.

GT-Lite reduces this effect by inserting the smoothing operator C. The convolution averages local neighborhoods in token space, acting as a low-pass filter on representation geometry. This regularization reduces the sensitivity of downstream sub-operators to small perturbations induced by upstream ones. Consequently, the pairwise commutator energies

$$E_{\mathrm{comm}}(A, C; x), \quad E_{\mathrm{comm}}(C, F; x), \quad E_{\mathrm{comm}}(A, F; x)$$

are all reduced relative to the baseline Transformer.

From a learning-dynamics perspective, the smoothing operator improves *exchangeability* of updates: the effective representation change induced by one sub-operator does not drastically alter how the others behave. This reduces gradient interference and improves training stability, even though the forward graph remains unchanged.

GeomDecoderBlock: extension to autoregressive decoding

Listing 2 shows the corresponding GT-Lite decoder block, which extends the same smoothing principle to autoregressive and cross-attention settings.

In the decoder, both causal self-attention and encoder–decoder cross-attention introduce global dependencies. The local smoothing operator again acts to reduce order sensitivity between these attention stages and the feed-forward network, improving dynamic compositionality in autoregressive generation.

SUMMARY. GT-Lite demonstrates that reducing commutator energy does not require complex geometric machinery. Even a simple local smoothing operator, when properly integrated into a residual architecture, can significantly improve compatibility between sub-operators. This provides a minimal, code-level illustration of how Diagrammatic Backpropagation probes dynamic compositionality and why geometric inductive biases matter for stable learning.

GT-Full Under the Hood: Simplicial Transport as Coordinate Alignment

GT-Full extends the GT-Lite architecture by replacing local convolutional smoothing with explicit geometric transport defined over a simplicial complex of token interactions. In contrast to GT-Lite, which enforces compatibility through local averaging, GT-Full introduces a structured mechanism that aligns representation geometry across tokens using simplicial message passing. This section unpacks the GT-Full encoder block line by line and explains why this alignment dramatically reduces commutator energy while preserving expressive global structure.

GeomFullEncoderBlockSeq: code structure

Listing 3 and Listing 4 show the implementation of the GT-Full encoder block. The block retains the standard Transformer attention and feed-forward stages, but inserts an additional geometric transport operator that acts on a graph (or simplicial complex) over token positions.

Sub-operators in GT-Full

From the perspective of dynamic compositionality, the GT-Full encoder block contains three interacting sub-operators:

- **Attention operator** A, defined as

$$A(x) = \mathrm{LN}_{attn}(x + \mathrm{Attn}(x)).$$

Listing 2: GT-Lite decoder block

```
class GeomDecoderBlock(nn.Module):
    def __init__(self, d_model: int, n_heads: int, dim_ff: int, dropout: float = 0.1):
        super().__init__()
        self.self_attn = nn.MultiheadAttention(
            d_model, n_heads, dropout=dropout, batch_first=True
        )
        self.cross_attn = nn.MultiheadAttention(
            d_model, n_heads, dropout=dropout, batch_first=True
        )
        self.conv = nn.Conv1d(d_model, d_model, kernel_size=3, padding=1)

        self.lin1 = nn.Linear(d_model, dim_ff)
        self.lin2 = nn.Linear(dim_ff, d_model)

        self.ln_self  = nn.LayerNorm(d_model)
        self.ln_cross = nn.LayerNorm(d_model)
        self.ln_conv  = nn.LayerNorm(d_model)
        self.ln_ff    = nn.LayerNorm(d_model)

        self.dropout = nn.Dropout(dropout)
        self.dropout_ff = nn.Dropout(dropout)

    def forward(self, x, enc_out, causal_mask):
        self_out, _ = self.self_attn(x, x, x, attn_mask=causal_mask, need_weights=False)
        x = self.ln_self(x + self.dropout(self_out))

        cross_out, _ = self.cross_attn(x, enc_out, enc_out, need_weights=False)
        x = self.ln_cross(x + self.dropout(cross_out))

        x_conv = self.conv(x.transpose(1, 2)).transpose(1, 2)
        x = self.ln_conv(x + 0.2 * x_conv)

        ff = self.lin2(self.dropout_ff(F.relu(self.lin1(x))))
        x = self.ln_ff(x + ff)
        return x
```

Listing 3: GT-Full encoder block

```
class GeomFullEncoderBlockSeq(nn.Module):
    def __init__(
        self,
        d_model: int,
        n_heads: int,
        dim_ff: int,
        max_len_src: int,
        num_rel: int = 1,
        dropout: float = 0.1,
        gt_depth: int = 1,
    ):
        super().__init__()
        self.self_attn = nn.MultiheadAttention(
            embed_dim=d_model,
            num_heads=n_heads,
            dropout=dropout,
            batch_first=True,
        )
        self.lin1 = nn.Linear(d_model, dim_ff)
        self.lin2 = nn.Linear(dim_ff, d_model)

        self.ln_attn = nn.LayerNorm(d_model)
        self.ln_ff   = nn.LayerNorm(d_model)

        self.dropout_attn = nn.Dropout(dropout)
        self.dropout_ff   = nn.Dropout(dropout)

        self.gt = GeometricTransformerV2(
            dim=d_model,
            depth=gt_depth,
            num_rel=num_rel,
        )

        # fixed positional graph over sequence indices
        src_indices, dst_indices = [], []
        for i in range(max_len_src - 1):
            src_indices.extend([i, i + 1])
            dst_indices.extend([i + 1, i])
        edge_index = torch.tensor([src_indices, dst_indices], dtype=torch.long)

        self.register_buffer("edge_index", edge_index)
        self.register_buffer("rel_ids", torch.zeros(edge_index.size(1), dtype=torch.long))
        self.register_buffer("dom_ids", torch.zeros(edge_index.size(1), dtype=torch.long))
```

Listing 4: GT-Full encoder block

```
class GeomFullEncoderBlockSeq(nn.Module):
    def __init__(
        self,
        d_model: int,
        n_heads: int,
        dim_ff: int,
        max_len_src: int,
        num_rel: int = 1,
        dropout: float = 0.1,
        gt_depth: int = 1,
    ):

...

    def forward(self, x: torch.Tensor) -> torch.Tensor:
        B, L, D = x.shape

        attn_out, _ = self.self_attn(x, x, x, need_weights=False)
        x = self.ln_attn(x + self.dropout_attn(attn_out))

        ff = self.lin2(self.dropout_ff(F.relu(self.lin1(x))))
        x = self.ln_ff(x + ff)

        # geometric transport
        h = x.reshape(B * L, D)
        edge_index = self.edge_index[:, self.edge_index[0] < L]
        rel_ids = self.rel_ids[self.edge_index[0] < L]
        dom_ids = self.dom_ids[self.edge_index[0] < L]
        h_ref = self.gt(h, edge_index, rel_ids, dom_ids)
        return h_ref.reshape(B, L, D)
```

- **Feed-forward operator** F, defined as

$$F(x) = \mathrm{LN}_{ff}(x + \mathrm{FF}(x)).$$

- **Geometric transport operator** G, defined by lifting token representations into a structured graph space, performing message passing, and reading back into the token space.

The overall block computes $G(F(A(x)))$. As before, the architecture fixes this order; the question addressed by the commutator-energy probe is how sensitive learning dynamics are to alternative local compositions of these operators.

What geometric transport actually does

The geometric transport operator G acts on the flattened representation $h \in \mathbb{R}^{(B\cdot L)\times d}$ and propagates information along a fixed positional graph defined by `edge_index`. Each node corresponds to a token position, and edges encode adjacency relations. Internally, the `GeometricTransformerV2` module performs multiple rounds of message passing, updating each token's representation based on its neighbors and relation embeddings.

Crucially, this process aligns representations across tokens by repeatedly exchanging information in a shared geometric coordinate system. Unlike local convolution, which enforces compatibility only within a small neighborhood, geometric transport spreads alignment globally across the sequence.

Why GT-Full reduces commutator energy

The commutator-energy probe for GT-Full evaluates

$$E_{\mathrm{comm}}(A, F; x), \quad E_{\mathrm{comm}}(A, G; x), \quad E_{\mathrm{comm}}(F, G; x).$$

Geometric transport reduces all three terms simultaneously. By aligning local coordinate frames, G ensures that the deformations induced by attention and feed-forward operators are expressed in a compatible geometric basis. As a result, applying A before G produces a similar effect to applying G before A, and likewise for F.

From a learning-dynamics perspective, G reduces curvature mismatch between sub-operators: small perturbations introduced by one stage do not drastically change the effective behavior of the others. This leads to low and stable commutator energy throughout training, as observed empirically in the v1 paper.

Preserving expressivity through alignment

An important distinction between GT-Lite and GT-Full is that GT-Full does not achieve low commutator energy by suppressing global structure. Instead, it introduces a geometric scaffold that allows expressive, long-range interactions to coexist with dynamic compositionality. This explains why GT-Full supports persistent global topological structure while maintaining stable learning dynamics.

In contrast, GT-Lite reduces order sensitivity primarily through local regularization, which can limit the emergence of rich global geometry. GT-Full achieves a stronger form of dynamic compositionality by explicitly coordinating sub-operator deformations across the entire sequence.

SUMMARY. GT-Full demonstrates that dynamic compositionality can be enforced through explicit geometric transport. By aligning representation geometry via simplicial message passing, GT-Full systematically reduces order sensitivity between attention, feed-forward, and transport operators, providing a mechanistic explanation for its superior commutator-energy behavior and empirical stability.

GT-MoE Under the Hood: Routing-Induced Order Sensitivity

The GT-MoE variant replaces the standard feed-forward sublayer with a mixture-of-experts (MoE) module whose effective transformation depends on learned routing decisions. While GT-MoE often achieves strong task-level performance and reduces commutator energy relative to a baseline Transformer, its mechanism differs fundamentally from that of GT-Full. In this section, we unpack the GT-MoE encoder block at the level of concrete code and explain how state-dependent routing alters dynamic compositionality.

GeomEncoderMoEBlock: code structure

Listing 5 shows the implementation of the GT-MoE encoder block. The block retains standard self-attention and residual structure, but replaces the feed-forward network with a gated mixture of experts.

Sub-operators in GT-MoE

From the perspective of dynamic compositionality, the GT-MoE block contains two primary sub-operators:

- **Attention operator** A, defined as
$$A(x) = \mathrm{LN}_{attn}(x + \mathrm{Attn}(x)).$$
- **Mixture-of-experts operator** M, defined as
$$M(x) = \mathrm{LN}_{ff}\left(x + \sum_{e=1}^{E} g_e(x)\, E_e(x)\right),$$
where $g_e(x)$ are learned gating probabilities and E_e are expert feed-forward networks.

Listing 5: GT-MoE encoder block

```
class GeomEncoderMoEBlock(nn.Module):
    def __init__(
        self,
        d_model: int,
        n_heads: int,
        dim_ff: int,
        n_experts: int = 4,
        dropout: float = 0.1,
        top_k: int = 2,
    ):
        super().__init__()
        self.self_attn = nn.MultiheadAttention(
            d_model, n_heads, dropout=dropout, batch_first=True
        )

        self.experts = nn.ModuleList([
            nn.Sequential(
                nn.Linear(d_model, dim_ff),
                nn.GELU(),
                nn.Linear(dim_ff, d_model),
            )
            for _ in range(n_experts)
        ])

        self.gate = nn.Linear(d_model, n_experts)

        self.ln_attn = nn.LayerNorm(d_model)
        self.ln_ff   = nn.LayerNorm(d_model)
        self.dropout = nn.Dropout(dropout)

    def forward(self, x: torch.Tensor) -> torch.Tensor:
        attn_out, _ = self.self_attn(x, x, x, need_weights=False)
        x = self.ln_attn(x + self.dropout(attn_out))

        gate_logits = self.gate(x)
        gate_probs = gate_logits.softmax(dim=-1)

        if self.top_k < gate_probs.size(-1):
            top_vals, top_idx = gate_probs.topk(self.top_k, dim=-1)
            sparse = torch.zeros_like(gate_probs)
            sparse.scatter_(2, top_idx, top_vals)
            gate_probs = sparse / (sparse.sum(dim=-1, keepdim=True) + 1e-8)

        ff_out = torch.zeros_like(x)
        for e_idx, expert in enumerate(self.experts):
            w_e = gate_probs[..., e_idx].unsqueeze(-1)
            ff_out = ff_out + w_e * expert(x)

        x = self.ln_ff(x + self.dropout(ff_out))
```

The overall block computes $M(A(x))$. Unlike the standard feed-forward network, the effective transformation implemented by M depends explicitly on routing decisions computed from the current representation.

Routing as state-dependent branching

The defining feature of GT-MoE is that routing decisions introduce conditional computation. For two nearby representations x and x', the set of active experts and their relative weights can differ substantially. As a result, the effective deformation of the representation space induced by M can change discontinuously as a function of x.

From the perspective of dynamic compositionality, this introduces an additional source of order sensitivity: applying attention before routing can change which experts are selected, while applying routing before attention would alter the attention input in a different way. The commutator-energy probe captures this effect by measuring

$$E_{\text{comm}}(A, M; x) = \|A(M(x)) - M(A(x))\|_2^2.$$

Why GT-MoE exhibits intermediate commutator energy

Empirically, GT-MoE tends to reduce commutator energy relative to a baseline Transformer but does not achieve the same level of reduction as GT-Full. This behavior follows directly from its mechanism. Mixture routing allows different experts to specialize and reduces interference within each expert, but it does not align representations across tokens or across experts in a shared geometric coordinate system. As a result, order sensitivity introduced by routing remains, especially when gating decisions are sharp or sparse.

In practice, GT-MoE occupies an intermediate regime: it improves dynamic compositionality by isolating incompatible deformations into different experts, but it does not eliminate order sensitivity arising from state-dependent branching. This explains the more variable commutator-energy trajectories observed for GT-MoE in the v1 experiments.

Comparison with GT-Lite and GT-Full

The contrast between GT-MoE and GT-Full highlights two distinct strategies for controlling dynamic compositionality:

- GT-Lite reduces order sensitivity by smoothing representations locally.
- GT-Full reduces order sensitivity by aligning representations globally through geometric transport.
- GT-MoE reduces interference by routing different representations to different experts, at the cost of introducing additional state-dependent branching.

Only GT-Full simultaneously achieves low commutator energy and supports coherent global structure. GT-MoE, while powerful, illustrates that conditional computation alone is insufficient to fully control dynamic compositionality.

SUMMARY. GT-MoE demonstrates that reducing commutator energy can be achieved through conditional computation as well as geometric alignment, but the two mechanisms have distinct dynamical consequences. By unpacking the MoE block at code level, we clarify why GT-MoE behaves differently from GT-Lite and GT-Full and why geometric transport plays a unique role in enforcing dynamic compositionality.

Diagrammatic Backpropagation as Čech-Style Descent

In this section we describe the algorithmic framework underlying diagrammatic backpropagation (DB), building on the detailed definitions of the GT models in the previous sections. Having analyzed dynamic compositionality at the level of individual encoder blocks, we now summarize how these components are orchestrated in practice. The following algorithms restate the DB+GT implementation in pseudocode form, emphasizing where commutator energy is measured, how it is optionally incorporated into training, and how geometric and conditional operators are invoked. Unlike the abstract presentation in the earlier paper, these algorithms are intended as a direct guide to the accompanying implementation.

Algorithm 6 is computed with `no_grad` and is used as an instrumental probe; DB training corresponds to adding one or more of these commutator terms (or higher-order generalizations) as auxiliary losses. In language-modeling and seq2seq experiments, we use Algorithm 7 with $\mathcal{L}_{\text{geom}} = 0$ and optionally include $\mathcal{L}_{\text{db}}$ (commutator-energy control) as an auxiliary loss. In diagrammatic benchmarks compiled into simplicial constraint instances, $\mathcal{L}_{\text{geom}}$ corresponds to triangle/square residual energies as in v1."

Algorithm 7 specifies the DB training loop, again written in a way that is faithful to the actual implementation.

Algorithm 8 defines the GT-Full Geometric Transport operator G used in the simplicial/graph message passing procedure. Algorithm 8 corresponds directly to the `GeomFullEncoderBlockSeq.forward()` transport path: the tensor z is flattened into h of shape (BL, d), the precomputed positional adjacency edge_index is masked to respect the current sequence length L, and GT applies K rounds of message passing and state updates. In our experiments, the positional graph is a bidirectional chain, but the same interface supports richer adjacency (e.g., dependency edges, retrieved links, or simplicial neighborhoods). Relation and domain ids provide typed edge features used by the message function ϕ_θ.

Algorithm 9 defines the local geometric smoothing operator C. Algorithm 9 corresponds to the GT-Lite smoothing stage in `GeomEncoderBlock` and `GeomDecoderBlock`. The convolution mixes each token with its immediate neighbors, acting as a local low-pass filter. The scaling factor α ensures smoothing is a gentle correction rather than a dominating transform, improving compatibility between attention and feed-forward updates.

Finally, Algorithm 10 defines the GT-MoE mixing operator M. Algorithm 10 corresponds to the MoE stage in `GeomEncoderMoEBlock`. The gating distribution p is computed per token and optionally sparsified via top-k routing. Expert outputs are then aggregated as a weighted sum. Because routing is state-dependent, small changes to z can change which experts dominate, introducing a distinct form of order sensitivity relative to GT-Lite and GT-Full.

From Čech Obstruction to Order Sensitivity: Reinterpreting v1 Results

The experimental results reported in the v1 paper established a striking and consistent empirical pattern: Geometric Transformer (GT) variants systematically reduce a proxy for local-to-global inconsistency—termed the Čech obstruction proxy—while a baseline Transformer does not. In this section, we reinterpret those findings through the lens of dynamic compositionality developed in Sections –, showing that the observed obstruction trajectories are a direct consequence of order sensitivity between learned sub-operators inside Transformer blocks.

Reframing the obstruction proxy

In the v1 paper, the Čech obstruction proxy was introduced as a metricized residual over local overlap patterns (triangles and squares) in a computational diagram. Operationally, this proxy was computed as the average of normalized commutator energies between pairs of sub-operators within each encoder block. Although motivated using descent and gluing language, the proxy itself is a purely forward-pass measurement:

$$\mathrm{Comm}(T_1, T_2; x) = \frac{\|T_1(T_2(x)) - T_2(T_1(x))\|_2^2}{\|x\|_2^2 + \varepsilon}.$$

Sections – showed that this quantity can be understood more concretely as a measure of *order sensitivity* between learned sub-operator deformations. High commutator energy indicates that applying one sub-operator significantly alters the effective behavior of another, making learning dynamics sensitive to update order. Low commutator energy indicates approximate exchangeability of updates and improved dynamic compositionality.

Why obstruction grows in baseline Transformers

In the v1 WikiText-103 experiments, the baseline Transformer exhibited steadily increasing obstruction over training, coinciding with degraded validation performance. From the dynamic compositionality perspective, this behavior is expected. As depth increases, repeated interactions between global attention operators and local nonlinear feed-forward networks amplify order sensitivity. Small mismatches in learned deformations accumulate across layers, leading to increasing gradient interference and unstable optimization.

The obstruction proxy thus tracks a failure mode intrinsic to vanilla Transformer learning dynamics: although the forward graph is statically compositional, the learned updates induced by different sub-operators are poorly coordinated on the representation manifold visited during training.

GT-Lite: partial mitigation via local smoothing

GeomTrans-Lite consistently reduced obstruction early in training but exhibited gradual obstruction drift at longer horizons. This behavior follows directly from its mechanism. The local geometric smoothing operator introduced in GT-Lite acts as a low-pass filter on representations, reducing sharp state dependence and improving compatibility between attention and feed-forward updates. However, because smoothing is local and does not align representations globally, order sensitivity can re-emerge as depth and task complexity increase.

Thus, GT-Lite improves dynamic compositionality relative to a baseline Transformer, but does not fully control long-range accumulation of order sensitivity.

GT-Full: enforcing dynamic compositionality through geometry

In contrast, the v1 experiments showed that GeomTrans-Full maintains low and remarkably stable obstruction throughout training across depths and scales. Sections and explain why this behavior is structurally inevitable. By introducing explicit geometric transport via simplicial message passing, GT-Full aligns representation geometry across tokens and layers. This alignment reduces curvature mismatch between learned deformations induced by attention and feed-forward operators, ensuring that their interactions remain approximately exchangeable.

From this perspective, the stability of the obstruction proxy in GT-Full is not a secondary effect, but a direct manifestation of architectural control over dynamic compositionality.

GT-MoE: conditional computation and intermediate regimes

The v1 results for GT-MoE revealed intermediate obstruction behavior: lower than the baseline Transformer, but higher and more variable than GT-Full.

This is also consistent with the mechanistic analysis. Mixture-of-experts routing isolates incompatible deformations into separate experts, reducing direct interference. However, routing decisions are themselves state-dependent, introducing branching behavior that preserves some degree of order sensitivity.

GT-MoE therefore occupies a distinct regime: it improves learning dynamics by conditional specialization, but does not fully eliminate order sensitivity in the way geometric alignment does.

SUMMARY. Viewed through the lens of dynamic compositionality, the empirical findings of the v1 paper admit a unified explanation. The Čech obstruction proxy measures order sensitivity between learned sub-operators. Baseline Transformers accumulate this order sensitivity with depth, GT-Lite mitigates it locally, GT-MoE mitigates it conditionally, and GT-Full suppresses it globally through explicit geometric transport. This reinterpretation completes the explanatory arc from the instrumental probes introduced in v1 to the code-level mechanisms analyzed in this paper.

Conclusion: Dynamic Compositionality as a First-Class Design Principle

This paper set out to explain, at a code- and mechanism-level, the empirical phenomena reported in the v1 Diagrammatic Backpropagation (DB) and Geometric Transformer (GT) study. Our central claim is that the observed behavior of GT models can be understood through a single unifying concept: *dynamic compositionality*. While Transformer architectures are statically compositional by construction, their learning dynamics under gradient descent are not guaranteed to preserve compatibility between interacting sub-operators. The resulting order sensitivity gives rise to gradient interference, instability, and degraded generalization.

We showed that the commutator-energy probe introduced in v1 is a concrete, forward-only measurement of this order sensitivity. Despite its simplicity, this probe captures a fundamental failure mode of deep learning systems: learned sub-operators can act as poorly coordinated deformations of a shared representation space. When such incompatibilities accumulate across depth and training time, optimization becomes fragile. The v2 analysis demonstrated that this behavior arises naturally even in minimal residual MLPs and becomes amplified in Transformer-scale architectures.

By unpacking the actual DB+GT implementation line by line, we clarified how different architectural choices control dynamic compositionality in distinct ways. GT-Lite reduces order sensitivity through local geometric smoothing, which regularizes representations but does not fully prevent long-range accumulation of incompatibility. GT-MoE mitigates interference through conditional routing, isolating deformations into specialized experts while introducing additional state-dependent branching. GT-Full achieves the strongest and most stable reduction in commutator energy by aligning representation geometry globally via explicit geometric transport. This alignment allows expressive global structure to coexist with stable learning dynamics, explaining the distinctive empirical signature of GT-Full observed in v1.

From a broader perspective, Diagrammatic Backpropagation can be viewed as a control mechanism for learning-time compositionality. Rather than assuming functoriality or commutativity of sub-operators, DB measures the degree to which learned updates are exchangeable on the data manifold and penalizes destructive order sensitivity. This reframes backpropagation not merely as a gradient computation, but as a coordination problem among interacting modules. The simplicity of the commutator-energy probe makes it a practical diagnostic for debugging architectures, comparing inductive biases, and guiding the design of new model components.

Finally, this work suggests a shift in how we reason about deep learning architectures. Expressivity alone is insufficient; the ability of a model to *learn coherently* is equally critical. Dynamic compositionality provides a language for articulating this requirement, and DB+GT offers a concrete instantiation of how it can be measured and enforced in practice. We hope that this handbook-style exposition will enable others to extend, adapt, and build upon DB+GT, and that future architectures will increasingly treat learning-time compositionality as a first-class design principle.

Summary and Future Work

In this chapter, we introduced the concept of *dynamic composition*, a way to instrument the internal workings of deep learning networks such as Transformers. We showed that our commutator energy metric was reduced significantly more by Geometric Transformer models than by regular Transformers. We also dove into the actual `Python` code to see how GTs work, and ascribed the improved performance to the specific types of simplicial message passing that occurs in the geometric layer.

One can go far deeper in the analysis of GT models, specifically as it relates to their *mean-field behavior*, that is, as the width of the models is increased to ∞, we can model their asymptotic behavior. This type of analysis reveals exactly what GTs optimize that regular Transformer models are unable to do. We turn to this analysis in the next chapter.

Algorithm 6: Layerwise Commutator-Energy Instrumentation (Čech Obstruction Proxy)

Input: Model M with encoder layers $\{L_\ell\}_{\ell=1}^{L}$; input tokens $s \in \mathbb{N}^{B\times T}$; small constant $\varepsilon > 0$

Output: Scalar obstruction score $\mathrm{Obs}(M; s)$ and per-layer scores $\{\mathrm{Obs}_\ell\}_{\ell=1}^{L}$

```
1:  Set $M$ to eval mode; save previous training flag.
2:  $x \leftarrow \mathrm{PosEnc}(\mathrm{TokEmb}(s))$    ▷ $x \in \mathbb{R}^{B\times T\times d}$
3:  $\mathrm{ObsList} \leftarrow [\,]$
4:  for $\ell = 1$ to $L$ do
5:      $x_{\mathrm{in}} \leftarrow x$
6:      $\mathrm{denom} \leftarrow \mathrm{MSE}(x_{\mathrm{in}}) + \varepsilon$    ▷ scale normalization

7:      // Define sub-operators as in the actual forward block (including residual & LN).
8:      if $L_\ell$ is a baseline Transformer encoder block then
9:          $A(z) \leftarrow \mathrm{LN}_1(z + \mathrm{Drop}(\mathrm{Attn}(z)))$
10:         $F(z) \leftarrow \mathrm{LN}_2(z + \mathrm{FF}(z))$
11:         $\mathrm{Obs}_\ell \leftarrow \frac{\mathrm{MSE}(A(F(x_{\mathrm{in}}))-F(A(x_{\mathrm{in}})))}{\mathrm{denom}}$
12:         $x \leftarrow L_\ell(x)$
13:     else if $L_\ell$ is GT-Lite (attention + conv smoothing + FF) then
14:         $A(z) \leftarrow \mathrm{LN}_{attn}(z + \mathrm{Drop}(\mathrm{Attn}(z)))$
15:         $C(z) \leftarrow \mathrm{LN}_{conv}(z + \alpha \cdot \mathrm{Conv}(z))$    ▷ $\alpha \approx 0.2$
16:         $F(z) \leftarrow \mathrm{LN}_{ff}(z + \mathrm{FF}(z))$
17:         $\mathrm{Obs}_\ell \leftarrow \frac{1}{3\,\mathrm{denom}}\Big(\mathrm{MSE}(A(C(x_{\mathrm{in}})) - C(A(x_{\mathrm{in}}))) + \mathrm{MSE}(C(F(x_{\mathrm{in}})) - F(C(x_{\mathrm{in}}))) + \mathrm{MSE}(A(F(x_{\mathrm{in}})) - F(A(x_{\mathrm{in}})))\Big)$
18:         $x \leftarrow L_\ell(x)$
19:     else if $L_\ell$ is GT-Full (attention + FF + geometric transport) then
20:         $A(z) \leftarrow \mathrm{LN}_{attn}(z + \mathrm{Drop}(\mathrm{Attn}(z)))$
21:         $F(z) \leftarrow \mathrm{LN}_{ff}(z + \mathrm{FF}(z))$
22:         $G(z) \leftarrow \mathrm{GeomTransport}(z)$    ▷ flatten → message pass → reshape
23:         $\mathrm{Obs}_\ell \leftarrow \frac{1}{3\,\mathrm{denom}}\Big(\mathrm{MSE}(A(F(x_{\mathrm{in}})) - F(A(x_{\mathrm{in}}))) + \mathrm{MSE}(F(G(x_{\mathrm{in}})) - G(F(x_{\mathrm{in}}))) + \mathrm{MSE}(A(G(x_{\mathrm{in}})) - G(A(x_{\mathrm{in}})))\Big)$
24:         $x \leftarrow L_\ell(x)$
25:     else if $L_\ell$ is GT-MoE (attention + MoE mixing) then
26:         $A(z) \leftarrow \mathrm{LN}_{attn}(z + \mathrm{Drop}(\mathrm{Attn}(z)))$
27:         $M(z) \leftarrow \mathrm{LN}_{ff}(z + \mathrm{MoE}(z))$    ▷ gating + top-$k$ + experts
28:         $\mathrm{Obs}_\ell \leftarrow \frac{\mathrm{MSE}(A(M(x_{\mathrm{in}}))-M(A(x_{\mathrm{in}})))}{\mathrm{denom}}$
29:         $x \leftarrow L_\ell(x)$
30:     else
31:         $\mathrm{Obs}_\ell \leftarrow \mathrm{NaN}$;    $x \leftarrow L_\ell(x)$
32:     end if
33:     $\mathrm{ObsList.append}(\mathrm{Obs}_\ell)$
34: end for
35: $\mathrm{Obs}(M; s) \leftarrow \mathrm{mean}\big(\{\mathrm{Obs}_\ell : \mathrm{Obs}_\ell \text{ is finite}\}\big)$
36: Restore training mode; return $\mathrm{Obs}(M; s), \mathrm{ObsList}$.
```

Algorithm 7: Diagrammatic Backpropagation Training Loop (code-faithful)

Input: Model M_θ (Transformer or GT variant); dataset iterator $\mathcal{B}$ yielding minibatches (s, y); optimizer AdamW; steps T; warmup steps T_w; weights λ_{db}, λ_{geom}; clipping norm c; small constant ε

Output: Trained parameters θ

```
1:  **Initialize:** set global step $t \leftarrow 0$
2:  **for** $t = 1$ **to** $T$ **do**
3:      Sample minibatch $(s, y) \sim \mathcal{B}$    ▷ $s$: tokens (or inputs), $y$: labels/targets

4:      **// 1) Forward pass for the primary task**
5:      $\hat{y} \leftarrow M_\theta(s)$
6:      $\mathcal{L}_{\text{task}} \leftarrow \text{LossTask}(\hat{y}, y)$    ▷ e.g., cross-entropy for LM

7:      **// 2) Optional DB control loss (commutator-energy terms)**
8:      $\mathcal{L}_{\text{db}} \leftarrow 0$
9:      **if** $\lambda_{\text{db}} > 0$ **then**
10:         $(\text{Obs}, \{\text{Obs}_\ell\}) \leftarrow \textsc{CommProbe}(M_\theta, s; \varepsilon)$    ▷ Algorithm 6
11:         $\mathcal{L}_{\text{db}} \leftarrow \text{Obs}$    ▷ scalar average over layers; can also weight layers/heads
12:     **end if**

13:     **// 3) Optional diagrammatic curvature loss (triangle/square residuals)**
14:     $\mathcal{L}_{\text{geom}} \leftarrow 0$
15:     **if** $\lambda_{\text{geom}} > 0$ **and** $(s, y)$ includes a compiled diagram/simplicial instance **then**
16:             ▷ For diagram benchmarks, $s$ contains $(\mathcal{D}, x)$ with constraints.
17:         $\mathcal{L}_{\text{geom}} \leftarrow \textsc{CurvatureEnergy}(M_\theta, s)$    ▷ triangle/square residual energy; cf. v1 Alg. (learned energy)
18:     **end if**

19:     **// 4) Warmup schedule for auxiliary terms (optional but common)**
20:     $\lambda_t^{\text{db}} \leftarrow \lambda_{\text{db}} \cdot \min(1,\ t/T_w)$
21:     $\lambda_t^{\text{geom}} \leftarrow \lambda_{\text{geom}} \cdot \min(1,\ t/T_w)$

22:     **// 5) Total loss and parameter update**
23:     $\mathcal{L} \leftarrow \mathcal{L}_{\text{task}} + \lambda_t^{\text{db}} \mathcal{L}_{\text{db}} + \lambda_t^{\text{geom}} \mathcal{L}_{\text{geom}}$
24:     $\theta \leftarrow \theta - \eta \nabla_\theta \mathcal{L}$    ▷ implemented via AdamW step
25:     Clip gradients: $\|\nabla_\theta \mathcal{L}\| \leq c$
26:     Optional: normalize selected parameter groups (e.g., edge transforms / relation embeddings)
27: **end for**
28: **return** $\theta$
```

Algorithm 8: GT-Full Geometric Transport Operator G (Simplicial/Graph Message Passing)

```
Input: Token states $z \in \mathbb{R}^{B\times L\times d}$; fixed adjacency edge_index$\in \mathbb{N}^{2\times E_{\max}}$ over positions $0,\ldots,L_{\max}-1$; relation ids rel_ids$\in \mathbb{N}^{E_{\max}}$; domain ids dom_ids$\in \mathbb{N}^{E_{\max}}$; depth $K$; learnable parameters $\theta_G$
Output: Transported token states $G(z) \in \mathbb{R}^{B\times L\times d}$
1: // 1) Flatten batch and restrict edges to current sequence length
2: $h \leftarrow \text{reshape}(z, (B{\cdot}L)\times d)$
3: $m \leftarrow \{e \in [1..E_{\max}] : \texttt{edge\_index}[0,e] < L\}$    ▷ mask endpoints within $L$
4: $E \leftarrow \texttt{edge\_index}[:,m]$; $r \leftarrow \texttt{rel\_ids}[m]$; $dID \leftarrow \texttt{dom\_ids}[m]$

5: // 2) Iterate $K$ rounds of geometric message passing
6: for $k = 1$ to $K$ do
7:     Initialize message accumulator $M \leftarrow 0 \in \mathbb{R}^{(BL)\times d}$
8:     for each directed edge $(u \leftarrow v)$ in $E$ do
9:         $e \leftarrow (u, v)$
10:        Edge embedding: $e_{\text{emb}} \leftarrow \text{Emb}_{rel}(r_e) + \text{Emb}_{dom}(dID_e)$
11:        Compute message: $m_{v\to u} \leftarrow \phi_\theta(h_v, h_u, e_{\text{emb}})$    ▷ e.g., MLP on $(h_v, h_u, e_{\text{emb}})$ or attention-style score
12:        $M_u \leftarrow M_u + m_{v\to u}$
13:    end for
14:    Degree normalize: $M \leftarrow \text{DegNorm}(M, E)$
15:    Update rule: $h \leftarrow \text{Update}_\theta(h, M)$    ▷ e.g., GRUCell/residual MLP + LayerNorm
16: end for

17: // 3) Reshape back to token grid
18: $G(z) \leftarrow \text{reshape}(h, B\times L\times d)$
19: return $G(z)$
```

Algorithm 9: GT-Lite Local Geometric Smoothing Operator C (1D Conv + Residual + LN)

Input: Token states $z \in \mathbb{R}^{B \times L \times d}$; convolution weights θ_C; smoothing scale α (e.g., 0.2); LayerNorm parameters θ_{LN}

Output: Smoothed token states $C(z) \in \mathbb{R}^{B \times L \times d}$

```
1: // 1) Convert to Conv1d layout and apply local neighborhood mixing
2: $z^\top \leftarrow \text{transpose}(z,\ (B, L, d) \to (B, d, L))$
3: $u^\top \leftarrow \text{Conv1D}_{\theta_C}(z^\top)$ ▷ kernel size 3, padding 1, output shape $(B, d, L)$
4: $u \leftarrow \text{transpose}(u^\top,\ (B, d, L) \to (B, L, d))$

5: // 2) Residual update and normalization
6: $z' \leftarrow z + \alpha \cdot u$
7: $C(z) \leftarrow \text{LayerNorm}_{\theta_{\text{LN}}}(z')$
8: return $C(z)$
```

Algorithm 10: GT-MoE Mixing Operator M (Gating + Top-k + Expert Aggregation)

Input: Token states $z \in \mathbb{R}^{B \times L \times d}$; experts $\{E_e\}_{e=1}^{E}$; gating network $g_\theta : \mathbb{R}^d \to \mathbb{R}^E$; optional top-$k$ value k; LayerNorm parameters θ_{LN}

Output: Mixed token states $M(z) \in \mathbb{R}^{B \times L \times d}$

```
1: // 1) Compute gating probabilities per token
2: $\ell \leftarrow g_\theta(z)$                          ▷ $\ell \in \mathbb{R}^{B \times L \times E}$ (gate logits)
3: $p \leftarrow \text{softmax}(\ell,\ \text{over experts})$   ▷ $p \in \mathbb{R}^{B \times L \times E}$

4: // 2) Optional sparsification (top-$k$ routing) and renormalization
5: if $k < E$ then
6:     $(v, I) \leftarrow \text{TopK}(p, k)$              ▷ $v, I \in \mathbb{R}^{B \times L \times k}$
7:     $\tilde{p} \leftarrow 0$                          ▷ initialize $\tilde{p} \in \mathbb{R}^{B \times L \times E}$
8:     $\tilde{p}[\cdot, \cdot, I] \leftarrow v$            ▷ scatter top-$k$ weights into expert slots
9:     $p \leftarrow \tilde{p} / \left( \sum_{e=1}^{E} \tilde{p}_e + \epsilon \right)$   ▷ renormalize per token
10: end if

11: // 3) Apply experts and aggregate weighted outputs
12: $u \leftarrow 0 \in \mathbb{R}^{B \times L \times d}$
13: for $e = 1$ to $E$ do
14:     $u \leftarrow u + p_e \odot E_e(z)$    ▷ $p_e$ broadcast to shape $(B, L, d)$; $\odot$ elementwise
15: end for

16: // 4) Residual update and normalization
17: $z' \leftarrow z + u$
18: $M(z) \leftarrow \text{LayerNorm}_{\theta_{\text{LN}}}(z')$
19: return $M(z)$
```

Mean-Field Theory of Geometric Transformers

We introduce commutator energy—implemented as a scale-invariant delta-commutator ratio—as a quantitative order parameter for *learning-time compositionality* in deep neural networks. Empirically, this obstruction-style diagnostic predicts optimization behavior across Transformer architectures, sharply distinguishing PostLNResidual, Pre-LN, and Geometric Transformer (GT) variants. We develop a mean-field analysis showing how normalization placement, residual structure, and geometric alignment mechanisms control the expected non-commutativity of learned patch updates. We further connect GT transport to harmonic analysis on token geometry: simplicial message passing admits a Laplacian/diffusion interpretation that suppresses high-frequency variation in the residual stream. This motivates a complementary spectral diagnostic based on Dirichlet energy, which we log alongside commutator obstruction and persistent-homology probes of attention geometry. In data-limited WikiText-103 and WikiText-2 seq2seq regimes (1000 training sentences, 1000 optimization steps), GT variants achieve substantially lower validation cross-entropy/perplexity and $\sim$ 4–6$\times$ higher token accuracy than both Pre-LN and Post-LN Transformer baselines, while exhibiting reduced obstruction and lower encoder-output Dirichlet energy. Together, the theory and instrumentation provide a principled framework for designing compositional architectures whose internal operators remain compatible under training.

Introduction

Modern neural architectures, such as the Transformer, are explicitly compositional: they are built by stacking reusable sub-operators (attention, feed-forward maps, normalization layers, residual connections) into blocks and then composing blocks into deep networks. This *static* compositionality is visible in the forward computation graph. Yet, the learning dynamics induced by gradient descent need not respect this modular structure. In practice, training can become highly sensitive to how sub-operator updates interact on the representations visited during optimization, leading to unstable learning, brittle depth/context scaling, and poor sample efficiency.

DYNAMIC COMPOSITIONALITY AND COMMUTATOR ENERGY. We call a model *dynamically compositional* if the learning-induced deformations of its sub-operators are mutually compatible on the training trajectory, so that training is not overly sensitive to the effective order in which sub-operator updates interact. We operationalize failure of dynamic compositionality using a forward-pass diagnostic based on commutators. Given two patch maps T_1, T_2 acting on a shared residual stream x, we measure order sensitivity via commutator energy. In practice we use a scale-invariant *delta-commutator ratio* that removes the identity stream:

$$\begin{aligned} \mathrm{Comm}_\Delta(T_1, T_2; x) &= \frac{\|\Delta_{12}(x) - \Delta_{21}(x)\|_2^2}{\|\Delta_{12}(x)\|_2^2 + \|\Delta_{21}(x)\|_2^2 + \varepsilon}, && (2)\\ \Delta_{12}(x) &= T_1(T_2(x)) - x, && (3)\\ \Delta_{21}(x) &= T_2(T_1(x)) - x. && (4) \end{aligned}$$

Low commutator energy indicates approximate exchangeability of local updates on the representations encountered during training; high commutator energy indicates order-sensitive interference.

GEOMETRIC TRANSFORMERS AND A SPECTRAL VIEW OF TRANSPORT. Diagrammatic Backpropagation (DB) and a family of Geometric Transformers (GTs) were introduced in previous chapters. Beyond task-level metrics, we introduced instrumental probes combining algebraic diagnostics inspired by Čech cohomology [55], enabling direct measurement of (i) local operator compatibility and (ii) global attention geometry. A code-level mechanisms analysis was given in the previous chapter. In this chapter, we unify and extend these ideas with a mean-field theory of commutator energy across normalization regimes (PostLNResidual vs. Pre-LN) and GT alignment mechanisms.

[55] Raoul Bott and Loring W. Tu. *Differential Forms in Algebraic Topology*. Springer, 1982; and Robert Ghrist. *Elementary Applied Topology*. Createspace, 2014

A central new ingredient is a *harmonic-analysis* interpretation of GT-Full: simplicial message passing admits a Laplacian/diffusion viewpoint on a token graph, so transport acts as a learned low-pass filter that suppresses high-frequency variation in the residual stream. This motivates a complementary spectral diagnostic: the *Dirichlet energy* of encoder representations on the token graph. Together, commutator obstruction (order sensitivity) and Dirichlet energy (spectral roughness) provide a practical bridge between architectural design and learning dynamics.

Čech-style obstruction proxy

Performance curves alone do not explain *why* some architectures learn reliably and sample-efficiently while others plateau or become unstable. We therefore introduce a computable proxy for local-to-global inconsistency. We treat each encoder block as a collection of local “patch maps” (e.g., attention A, feed-forward F, and in GT variants an additional geometric operator C or transport operator G_Δ) acting on the shared residual stream. The layer-level *Čech obstruction proxy* is the average delta-commutator ratio over relevant patch pairs; the model-level proxy is the average over encoder layers. The metrics we use for each model are summarized in the table below, computed in evaluation mode to remove stochastic effects from dropout. This proxy is a metricized Čech-1 inconsistency: it measures whether locally defined updates can be glued into a globally coherent transformation on the overlap interface.

Model	**Commutator Energy**
Transformer	average of $\mathrm{Comm}_\Delta(A, F; x)$
GT-Lite	average of $\mathrm{Comm}_\Delta(A, C; x), \mathrm{Comm}_\Delta(C, F; x), \mathrm{Comm}_\Delta(A, F; x)$
GT-Full	average of $\mathrm{Comm}_\Delta(A, F; x), \mathrm{Comm}_\Delta(F, G_\Delta; x), \mathrm{Comm}_\Delta(A, G_\Delta; x)$

Table 3: Commutator Energy Metrics for Different Transformer Models.

Why a spectral diagnostic: Dirichlet energy on token geometry

The Laplacian view of GT suggests that geometric transport should reduce high-frequency variation of representations over the token graph. We therefore log a simple Dirichlet-energy diagnostic at the encoder output:

$$\mathcal{E}_L(V) = \mathrm{tr}(V^\top L V) = \frac{1}{2} \sum_{(u,v)\in E} \|V_u - V_v\|_2^2,$$

normalized to be comparable across length and width. Low Dirichlet energy indicates smoothness on token geometry (a diffusion/low-pass signature), while high Dirichlet energy indicates roughness and potential instability. In the data-limited regimes studied in this paper, Dirichlet energy complements the commutator obstruction proxy by isolating spectral roughness effects from operator-order interference.

A Minimal Demonstration: Residual MLPs on Two Moons

To illustrate that commutator energy is not specific to Transformers, we present a minimal demonstration using a residual MLP trained on the two-moons task. Although all models reach perfect classification accuracy, their internal dynamics differ sharply: the baseline residual MLP exhibits increasing commutator energy during training, while GT-Lite maintains substantially lower obstruction and GT-Full achieves the lowest and most stable commutator energy. This example highlights that dynamic compositionality failure is a general phenomenon of gradient-based learning in compositional architectures, and that geometric alignment mechanisms can mitigate it even when task metrics saturate.

Empirical preview: data-limited regimes reveal clear separation

A recurring theme in this paper is that *data-limited* regimes expose architecture-dependent differences that are not visible when data and compute are abundant. In controlled experiments on WikiText-103 and WikiText-2 prefix$\to$suffix seq2seq tasks with $n_{\text{train}} = 1000$ sentences and a fixed optimization budget (1000 steps), GT variants achieve dramatically lower validation cross-entropy/perplexity and $\sim$ 4–6$\times$ higher token accuracy than both PostLNResidual and Pre-LN Transformers. These gains co-occur with reduced Čech obstruction and lower encoder-output Dirichlet energy, providing direct empirical support for the theory: geometric transport behaves like a learned diffusion regularizer on token geometry that stabilizes the composition of local updates.

Results: obstruction trajectories explain stability

Obstruction trajectories provide a mechanistic explanation for stability and sample efficiency. Baseline Transformers can accumulate increasing obstruction over training, indicating growing loop inconsistency among local updates. GT-Lite often reduces obstruction early but can exhibit drift; GT-Full maintains low and more stable obstruction by explicit transport. These patterns align with the mean-field theory developed in later sections and with the spectral/Dirichlet view: transport suppresses high-frequency instability modes on token geometry, reducing order sensitivity between patch updates and improving learning dynamics.

EMPIRICAL PREVIEW. Rather than emphasizing long-run benchmark curves, we focus on controlled, data-limited regimes in which architectural differences are amplified and can be mechanistically interpreted. Sections and summarize results on WikiText-103 and WikiText-2 with matched budgets and full instrumentation (task metrics, Čech obstruction, topology probes, and Dirichlet energy), providing a compact empirical anchor for the mean-field and spectral analyses developed later.

Model and Assumptions

Our goal is to analyze the learning-time non-commutativity of compositional neural network blocks through the lens of commutator energy. To this end, we adopt a simplified but standard mean-field setting that allows explicit calculation and asymptotic reasoning, while remaining faithful to the architectures studied empirically in earlier sections.

We use a scale-invariant *delta-commutator ratio* that removes the identity stream and normalizes by the residual update magnitudes:

$$\mathrm{Comm}_{\Delta}(T_1, T_2; x) = \frac{\|\Delta_{12}(x) - \Delta_{21}(x)\|_2^2}{\|\Delta_{12}(x)\|_2^2 + \|\Delta_{21}(x)\|_2^2 + \varepsilon}, \tag{5}$$

$$\Delta_{12}(x) = T_1(T_2(x)) - x, \; \Delta_{21}(x) = T_2(T_1(x)) - x. \tag{6}$$

This ratio is invariant to global rescaling of the residual updates and avoids domination by the shared identity path.

Residual Composition Model

We consider neural networks composed of residual blocks acting on representations $x \in \mathbb{R}^d$. Each block defines a transformation of the form

$$T_i(x) \;=\; x + \Delta_i(x), \tag{7}$$

where $\Delta_i \;:\; \mathbb{R}^d \;\to\; \mathbb{R}^d$ is a learned nonlinear map. This formulation encompasses residual multilayer perceptrons, Transformer attention and feed-forward sublayers, and Geometric Transformer (GT) components used throughout this work.

Given two such blocks T_i and T_j, we study the discrepancy between their two possible compositions:

$$T_i(T_j(x)) \quad \text{and} \quad T_j(T_i(x)). \tag{8}$$

Exact equality corresponds to commutativity of the two transformations on input x. In practice, learned blocks need only commute approximately; the degree of non-commutativity is quantified by the commutator energy.

Random Feature Mean-Field Setting

We analyze commutator energy in the infinite-width limit under standard mean-field assumptions. Specifically, we assume:

(A1) WIDTH SCALING. The representation dimension $d \;\to\; \infty$, while depth remains fixed. Weight matrices are scaled so that activations have $\mathcal{O}(1)$ variance.

(A2) RANDOM INITIALIZATION. Each Δ_i is parameterized by an independent random network at initialization. For concreteness, we consider

$$\Delta_i(x) \;=\; W_i\,\phi(x), \tag{9}$$

where $W_i \in \mathbb{R}^{d\times d}$ has i.i.d. entries with zero mean and variance σ^2/d, and $\phi(\cdot)$ is a pointwise nonlinearity such as ReLU or GELU. Different blocks $i \neq j$ have independent weights.

(A3) INPUT DISTRIBUTION. Inputs x are drawn from a distribution with bounded second moments, typically assumed isotropic in the mean-field limit. Expectations are taken with respect to both input randomness and weight initialization.

(A4) LAYER NORMALIZATION VARIANTS. We consider two normalization placements:

- **Post-LN:** normalization applied after residual addition,

$$T_i^{\text{Post}}(x) = \text{LN}(x + \Delta_i(x));$$

- **Pre-LN:** normalization applied before the nonlinear map,

$$T_i^{\mathrm{Pre}}(x) = x + \Delta_i(\mathrm{LN}(x)).$$

These correspond to the Post-LN and Pre-LN Transformer architectures.

(A4') BLOCK-LEVEL NORMALIZATION CONVENTIONS. We distinguish *PostLNResidual* blocks from the simplified "Post-LN" toy model. A block consists of K patch operators $\{T_k\}_{k=1}^{K}$ (e.g., attention, feed-forward, transport), acting on a shared residual stream.

- **PostLNResidual:** each patch applies LN after residual addition,

$$x \mapsto \mathrm{LN}\big(x + \Delta_k(x)\big), \qquad k = 1, \ldots, K.$$

- **PreLN:** each patch applies LN before its update and leaves the residual stream unnormalized within the block,

$$x \mapsto x + \Delta_k(\mathrm{LN}(x)), \qquad k = 1, \ldots, K,$$

  with an optional final normalization outside the block (as in standard PreLN Transformers).

This matches the implementations used in our GT-Lite, GT-Full, and GT-MoE experiments.

Local Linearization and Jacobian Structure

We will use T_i, T_j to denote either entire residual blocks or *patch maps* within a block (e.g., attention A, feed-forward F, smoothing C, transport G_Δ, or MoE mixing M), since our obstruction proxy is measured at the patch level.

To analyze commutator energy, we study the local linearization of each block. Let $J_i(x) = \nabla_x T_i(x)$ denote the Jacobian of T_i at x. For residual blocks of the form above,

$$J_i(x) \;=\; I + \nabla_x \Delta_i(x). \tag{10}$$

Under assumptions (A1)–(A3), the Jacobians $\nabla_x \Delta_i(x)$ converge in distribution to random matrices with controlled spectral statistics, as in classical mean-field analyses.

Using a first-order expansion, the commutator between two blocks can be approximated as

$$T_i(T_j(x)) - T_j(T_i(x)) \;\approx\; \big(J_i(x)J_j(x) - J_j(x)J_i(x)\big)\,\delta, \tag{11}$$

for small perturbations δ around x. This expression makes explicit that commutator energy measures the non-commutativity of Jacobians induced by distinct learned transformations.

Scope and Limitations

Our analysis focuses on initialization and early-training regimes where mean-field approximations are valid. We do not claim exact quantitative predictions for finite-width networks or late-stage training. Rather, the goal is to identify qualitative differences between architectures—Post-LN, Pre-LN, and GT-style alignment mechanisms—in terms of their expected commutator energy and its scaling with depth and normalization.

Empirical results demonstrate that these qualitative predictions persist well beyond the strict mean-field regime.

Expected Commutator Energy at Initialization

In this section we analyze the expected commutator energy between residual blocks at random initialization under the mean-field assumptions. The goal is to establish that nonzero commutator energy is a *generic* property of compositional residual networks, and to characterize how its magnitude depends on architectural choices such as normalization placement.

Definition of Commutator Energy

Given two residual blocks

$$T_i(x) = x + \Delta_i(x), \qquad T_j(x) = x + \Delta_j(x),$$

we define the commutator residual at input x as

$$R_{ij}(x) \;=\; T_i(T_j(x)) - T_j(T_i(x)). \tag{12}$$

The commutator energy is then given by

$$\mathcal{E}_{ij}(x) \;=\; \mathbb{E}\big[\|R_{ij}(x)\|_2^2\big], \tag{13}$$

where the expectation is taken over both input randomness and random initialization of the network parameters.

Exact commutativity corresponds to $\mathcal{E}_{ij}(x) \;=\; 0$. Our objective is to understand when $\mathcal{E}_{ij}(x)$ is generically nonzero and how it scales in the mean-field limit.

First-Order Expansion

Expanding the compositions explicitly yields

$$T_i(T_j(x)) = x + \Delta_j(x) + \Delta_i\big(x + \Delta_j(x)\big), \tag{14}$$

$$T_j(T_i(x)) = x + \Delta_i(x) + \Delta_j\big(x + \Delta_i(x)\big). \tag{15}$$

Subtracting gives

$$R_{ij}(x) = \Delta_i\big(x + \Delta_j(x)\big) - \Delta_j\big(x + \Delta_i(x)\big) + \Delta_j(x) - \Delta_i(x). \tag{16}$$

Under the mean-field assumption that $\Delta_i(x)$ and $\Delta_j(x)$ are $\mathcal{O}(1)$ but small relative to x in the large-width limit, we linearize Δ_i and Δ_j around x:

$$\Delta_i(x + \Delta_j(x)) \approx \Delta_i(x) + J_i(x)\,\Delta_j(x),$$

where $J_i(x) = \nabla_x \Delta_i(x)$ is the Jacobian.

Substituting and simplifying, first-order terms cancel and we obtain

$$R_{ij}(x) \;\approx\; J_i(x)\,\Delta_j(x) - J_j(x)\,\Delta_i(x). \tag{17}$$

This expression shows that commutator energy arises from the mismatch between how each block responds to the other's perturbation.

Jacobian Commutator Approximation

Using $\Delta_k(x) \approx J_k(x)\,x$ at initialization, Equation (17) can be further approximated as

$$R_{ij}(x) \;\approx\; \big(J_i(x)J_j(x) - J_j(x)J_i(x)\big)\,x. \tag{18}$$

Thus, up to first order, commutator energy measures the non-commutativity of the Jacobian matrices associated with distinct residual blocks.

Expected Non-Commutativity

Under assumptions (A1)–(A3), the Jacobians $J_i(x)$ and $J_j(x)$ are independent random matrices whose entries have zero mean and variance controlled by the activation function and weight scaling. For two independent random matrices drawn from such distributions, the expected Frobenius norm of their commutator is generically nonzero:

$$\mathbb{E}\big[\|J_iJ_j - J_jJ_i\|_F^2\big] > 0. \tag{19}$$

Moreover, this expectation increases with the variance of the Jacobian entries, which in turn depends on the variance of the input x and the non-linearity ϕ. As a consequence, even at initialization, compositional residual blocks are expected to exhibit nonzero commutator energy unless additional structure or constraints are imposed.

Implications

This analysis establishes that commutator energy is not an artifact of training, optimization, or specific architectures such as attention. Rather, it is a generic feature of composing independently parameterized residual transformations in high dimensions. Normalization and architectural alignment mechanisms influence commutator energy by modifying the statistics of the Jacobians $J_i(x)$ and their interactions.

In the next section, we show how Pre-LN normalization alters these statistics and provably reduces expected commutator energy relative to Post-LN architectures.

Effect of Layer Normalization on Expected Commutator Energy

We now analyze how layer normalization (LN) affects the expected commutator energy. Our goal is to show that LN modifies the statistics of the Jacobians entering the commutator term and thereby reduces expected non-commutativity, providing a theoretical explanation for the empirical stability of Pre-LN architectures in our experiments.

Layer Normalization as a Jacobian Rescaling Operator

Layer normalization maps an input vector $x \in \mathbb{R}^d$ to

$$\mathrm{LN}(x) = \frac{x - \mu(x)\mathbf{1}}{\sigma(x)}, \tag{20}$$

where $\mu(x)$ and $\sigma(x)$ denote the mean and standard deviation of the coordinates of x. In the mean-field limit $d \to \infty$, $\mu(x)$ concentrates around zero and $\sigma(x)$ concentrates around $\sqrt{\mathbb{E}[x_i^2]}$.

The Jacobian of LN has the form

$$J_{\mathrm{LN}}(x) = \frac{1}{\sigma(x)} \left(I - \frac{1}{d}\mathbf{1}\mathbf{1}^\top - \frac{(x-\mu(x)\mathbf{1})(x-\mu(x)\mathbf{1})^\top}{d\,\sigma(x)^2} \right), \tag{21}$$

which acts approximately as an isotropic rescaling operator on directions orthogonal to $\mathbf{1}$ in high dimensions. Crucially, LN suppresses variations in the magnitude of inputs entering subsequent nonlinear transformations.

Post-LN Residual Blocks

In Post-LN architectures, each residual block has the form

$$T_i^{\mathrm{Post}}(x) = \mathrm{LN}(x + \Delta_i(x)). \tag{22}$$

The Jacobian of this transformation is

$$J_i^{\mathrm{Post}}(x) = J_{\mathrm{LN}}(x + \Delta_i(x))\big(I + \nabla_x \Delta_i(x)\big). \tag{23}$$

Because normalization is applied after the residual addition, the Jacobian $\nabla_x \Delta_i(x)$ enters multiplicatively inside J_i^{Post}, retaining its full variance and directional structure. Consequently, the Jacobian commutator

$$J_i^{\mathrm{Post}} J_j^{\mathrm{Post}} - J_j^{\mathrm{Post}} J_i^{\mathrm{Post}}$$

inherits substantial variance from the unnormalized nonlinear maps, leading to large expected commutator energy.

This explains why Post-LN Transformers exhibit unstable learning dynamics and rapid accumulation of obstruction in practice.

Pre-LN Residual Blocks

In Pre-LN architectures, normalization is applied before each nonlinear map:

$$T_i^{\mathrm{Pre}}(x) = x + \Delta_i(\mathrm{LN}(x)). \tag{24}$$

The Jacobian now takes the form

$$J_i^{\mathrm{Pre}}(x) = I + \nabla_z \Delta_i(z)\, J_{\mathrm{LN}}(x), \qquad z = \mathrm{LN}(x). \tag{25}$$

Here, $J_{\mathrm{LN}}(x)$ acts as a variance-controlling operator that rescales the input to Δ_i before the nonlinear transformation is applied.

As a result, the effective Jacobians $\nabla_z \Delta_i(z) J_{\mathrm{LN}}(x)$ have reduced variance compared to the Post-LN case. Substituting into the commutator expression (18) yields

$$J_i^{\mathrm{Pre}} J_j^{\mathrm{Pre}} - J_j^{\mathrm{Pre}} J_i^{\mathrm{Pre}} = \tag{26}$$
$$\big(\nabla \Delta_i J_{\mathrm{LN}}\big)\big(\nabla \Delta_j J_{\mathrm{LN}}\big) - \tag{27}$$
$$\big(\nabla \Delta_j J_{\mathrm{LN}}\big)\big(\nabla \Delta_i J_{\mathrm{LN}}\big), \tag{28}$$

which is suppressed by the normalization-induced scaling of J_{LN}.

Qualitative Reduction of Expected Commutator Energy

Under mean-field assumptions, the variance reduction introduced by Pre-LN leads to a strict decrease in the expected commutator energy:

$$\mathbb{E}\big[\| J_i^{\mathrm{Pre}} J_j^{\mathrm{Pre}} - J_j^{\mathrm{Pre}} J_i^{\mathrm{Pre}} \|_F^2\big] < \tag{29}$$
$$\mathbb{E}\big[\| J_i^{\mathrm{Post}} J_j^{\mathrm{Post}} - J_j^{\mathrm{Post}} J_i^{\mathrm{Post}} \|_F^2\big]. \tag{30}$$

This provides a theoretical explanation for the improved stability of Pre-LN and observed empirically in our commutator-energy measurements.

However, normalization alone does not enforce alignment between the directions of distinct Jacobians. Consequently, Pre-LN reduces but does not eliminate expected non-commutativity, consistent with the residual commutator energy observed in practice.

Interpretation

From the perspective of commutator energy, Pre-LN normalization acts as a *magnitude control* mechanism: it rescales the inputs to each sub-operator and suppresses variance-driven instability. It does not, however, impose directional compatibility between learned transformations. In the next section, we show that Geometric Transformer architectures further reduce commutator energy by explicitly aligning sub-operator effects through geometric transport, providing a principled extension beyond normalization.

Geometric Alignment Beyond Normalization

We showed previously that Pre-LN architectures reduce expected commutator energy by controlling the magnitude of Jacobians entering residual blocks. However, empirical results consistently demonstrate that normalization alone does not eliminate learning-time non-commutativity. In this section, we argue that further reduction of commutator energy requires explicit control over the *directional structure* of learned transformations. Geometric Transformer (GT) architectures achieve this through alignment mechanisms that go beyond normalization.

Why Normalization Is Insufficient

Layer normalization rescales inputs to each nonlinear map, suppressing variance and stabilizing gradient flow. However, normalization acts isotropically in the space orthogonal to the all-ones vector. As a result, it does not constrain the relative orientations of Jacobians associated with distinct residual blocks.

In terms of the Jacobian commutator

$$J_i J_j - J_j J_i,$$

normalization reduces the magnitude of both J_i and J_j, but leaves their eigenvectors and principal directions unconstrained. Consequently, while Pre-LN reduces the overall scale of commutator energy, it cannot enforce approximate commutativity when J_i and J_j act in incompatible directions.

This observation explains why Pre-LN architectures exhibit improved stability but still accumulate nonzero commutator energy over training.

Geometric Alignment as Directional Control

To further suppress commutator energy, it is necessary to align the directions along which different residual blocks act. Geometric alignment refers to architectural mechanisms that encourage learned transformations to operate in compatible coordinate frames or along shared geometric structures.

Formally, consider the first-order commutator approximation

$$R_{ij}(x) \approx (J_i J_j - J_j J_i)\, x.$$

If J_i and J_j share approximately aligned eigenspaces, the commutator term is suppressed even when the individual Jacobians have nontrivial magnitude. Thus, alignment of Jacobian eigenspaces provides a distinct and complementary means of reducing commutator energy beyond variance control. Geometrically, this corresponds to reducing curvature induced by incompatible local coordinate frames on the representation manifold.

Geometric Transport in GT Architectures

GT architectures implement geometric alignment through explicit transport operators that propagate representations across a fixed relational or positional structure. In the simplest case (GT-Lite), local smoothing encourages nearby representations to evolve coherently, partially aligning the directions of subsequent transformations. In the full GT architecture, message passing over a graph or simplicial complex enforces consistency across overlapping neighborhoods of representations. Importantly, alignment does not require sub-operators to be equal or commute exactly; it only requires that their induced deformations act in compatible directions on the data manifold.

From a Jacobian perspective, geometric transport introduces coupling terms that bias $\nabla\Delta_i(x)$ toward acting along shared subspaces across blocks. This reduces directional mismatch between J_i and J_j, suppressing the commutator even when normalization alone is insufficient.

GT-FULL TRANSPORT AS A RESIDUAL UPDATE. In GT-Full, the geometric transport module is implemented as a residual message-passing stack. We therefore model it as a *delta transport* operator

$$\mathcal{G}_\Delta(x) \;=\; \mathrm{GT}(x) - x,$$

so that the block applies $x \mapsto x + \mathcal{G}_\Delta(\cdot)$ (PreLN) or $x \mapsto \mathrm{LN}(x + \mathcal{G}_\Delta(\cdot))$ (PostLNResidual). This avoids double-counting the identity stream in both analysis and instrumentation.

Mean-Field Interpretation of Alignment

In the mean-field limit, geometric alignment can be interpreted as reducing the variance of off-diagonal terms in the joint distribution of Jacobians. Whereas Pre-LN reduces the variance of individual Jacobians, GT-style alignment reduces their *cross-covariance*:

$$\mathrm{Cov}(J_i, J_j) \;\to\; \text{aligned}.$$

As a result, the expected Frobenius norm of the commutator decreases further:

$$\mathbb{E}\big[\|J_iJ_j - J_jJ_i\|_F^2\big] \;\ll\; \mathbb{E}\big[\|J_i^{\mathrm{Pre}}J_j^{\mathrm{Pre}} - J_j^{\mathrm{Pre}}J_i^{\mathrm{Pre}}\|_F^2\big].$$

This qualitative inequality aligns with empirical observations across residual MLPs, sequence-to-sequence models, and language modeling benchmarks.

Normalization vs. Alignment

Normalization alone cannot enforce approximate commutativity unless the dominant eigenspaces of distinct Jacobians are already aligned by chance. The distinction between normalization and alignment clarifies the relationship between Pre-LN Transformers and GT architectures:

- **Normalization** controls the scale of transformations and stabilizes gradient magnitudes.
- **Alignment** controls the directions of transformations and stabilizes their composition.

Pre-LN addresses the former, while GT addresses both.

Architectural Mechanisms for Alignment in Geometric Transformers

We showed that Pre-LN primarily reduces commutator energy by controlling the *magnitude* (variance) of Jacobians entering residual sublayers. We argued that further suppression requires *directional* control: reducing mismatch between the principal subspaces along which different sub-operators act. We now summarize, at an architectural level, how Geometric Transformer (GT) variants implement such alignment mechanisms.

OPERATOR DECOMPOSITION. We view a Transformer-style block as the composition of sub-operators acting on a shared residual stream. Let A denote the attention stage (including residual addition and normalization) and F the feed-forward stage. GT variants insert additional operators that alter the geometry of the representation space between A and F, yielding a block of the form

$$x \mapsto \cdots \circ F \circ \mathcal{G} \circ A(x),$$

where $\mathcal{G}$ is a geometric coordination operator. The commutator-energy probe evaluates order sensitivity between these operators (e.g., $E_{\mathrm{comm}}(A, F; x)$ and cross-terms involving $\mathcal{G}$), and thus directly measures whether learned deformations remain approximately exchangeable on the data manifold.

GT-LITE: LOCAL SMOOTHING AS LOW-PASS ALIGNMENT. In GT-Lite, $\mathcal{G} = C$ is a local smoothing operator (e.g., convolution or neighborhood mixing). Smoothing suppresses high-frequency variation in the residual stream, which reduces the sensitivity of downstream sub-operators to small upstream perturbations. In Jacobian terms, smoothing acts as a contraction on unstable directions, effectively reducing the contribution of rapidly varying eigenspaces to $\nabla\Delta(x)$. This partially aligns J_A and J_F by reducing directional mismatch in the components that cause large commutators. Empirically, this yields a substantial reduction in commutator energy relative to vanilla Transformers, but can exhibit slow drift as depth and task complexity increase.

GT-FULL: GEOMETRIC TRANSPORT AS COORDINATE-FRAME ALIGNMENT. In GT-Full, $\mathcal{G} = G$ is an explicit transport operator implemented via message passing over a fixed relational structure (graph or simplicial complex) on token positions. Transport couples neighboring representations and propagates information in a shared geometric coordinate system. This enforces consistency across overlapping neighborhoods and biases the learned deformations induced by A and F to act in compatible subspaces. At the Jacobian level, transport reduces cross-covariance mismatch between J_A and J_F by aligning their dominant eigenspaces across tokens, thereby suppressing $J_A J_F - J_F J_A$ beyond what normalization alone can achieve. This predicts the empirical signature that GT-Full maintains low and stable commutator energy while still supporting rich global structure (e.g., nontrivial topological signatures).

GT-MoE: CONDITIONAL SPECIALIZATION VS. GEOMETRIC ALIGNMENT. In GT-MoE, $\mathcal{G} = M$ is a mixture-of-experts mixing operator whose effective transformation depends on state-dependent routing. Routing can reduce interference by allowing different experts to specialize, but introduces additional branching sensitivity: small changes in the residual stream can alter expert selection and hence the effective Jacobian of M. Consequently, GT-MoE often occupies an intermediate regime: it improves dynamic compositionality relative to a baseline Transformer, but does not achieve the same degree of directional alignment as explicit transport. Thus GT-MoE mitigates interference by partitioning computation, whereas GT-Full mitigates interference by coordinating computation.

Lemma 2 (Commutator suppression by contractive transport)**.** *Assume a PreLN patch map $T_A(x) = x + \Delta_A(\mathrm{LN}(x))$ with $\|J_{\Delta_A}\| \le L_A$ on the support of training representations, and a transport patch $T_G(x) = x + \mathcal{G}_\Delta(\mathrm{LN}(x))$ where $\mathcal{G}_\Delta$ is (locally) α-Lipschitz with $\alpha < 1$. Then for any x,*

$$\|\Delta_{AG}(x) - \Delta_{GA}(x)\| \le c\, L_A\, \alpha\, \|J_{\mathrm{LN}}(x)\|\, \|\Delta_A(\mathrm{LN}(x))\|$$

for a constant c depending on the smoothness of the maps. Consequently, the delta-commutator ratio $\mathrm{Comm}_\Delta(T_A, T_G; x)$ is $O(\alpha^2)$ when α is small.

SUMMARY. Viewed through the lens of commutator energy, normalization primarily controls the *scale* of Jacobians, while GT mechanisms control their *relative orientation*. GT-Lite achieves partial alignment through local smoothing, GT-Full achieves stronger alignment through explicit transport, and GT-MoE achieves partial mitigation through conditional specialization. This architectural taxonomy explains the qualitative ordering observed empirically and motivates geometric alignment as a principled design objective for suppressing learning-time non-commutativity.

Implications

Geometric alignment provides a principled explanation for why GT architectures consistently achieve lower and more stable commutator energy than both Post-LN and Pre-LN baselines. More broadly, it suggests that architectural control of learning-time geometry is a necessary design principle for robust dynamic compositionality in deep networks. While we do not provide a complete closed-form mean-field computation for each GT mechanism, the mean-field/Jacobian view predicts the observed ordering of commutator energy trajectories across variants.

In the next section, we discuss the implications of commutator energy as a theoretical order parameter and outline directions for further analysis and architectural design.

Geometric Transport as Laplacian Smoothing and Harmonic Analysis

This section refines the mean-field view of Geometric Transformers (GT) by connecting simplicial message passing to graph Laplacians and spectral (harmonic) analysis on token geometry. The key idea is that GT transport acts as a learned diffusion operator on a token-position graph (or more generally a simplicial complex), functioning as a low-pass filter that suppresses high-frequency variation in the residual stream. This provides a principled mechanism for reduced order-sensitivity and improved sample-efficient learning, especially in data-limited regimes.

From message passing to a Laplacian step

Consider a fixed graph $G = (\mathcal{V}, \mathcal{E})$ on token positions with adjacency A and degree matrix D, and let $L = D - A$ denote the (unnormalized) graph Laplacian. For a node embedding matrix $V \in \mathbb{R}^{n\times d}$ (with $n = |\mathcal{V}|$ tokens and d hidden dimension), a prototypical linear message-passing update has the form

$$V \;\mapsto\; V + \Delta_G(V), \qquad (\Delta_G(V))_v \;=\; \sum_{u:(u,v)\in\mathcal{E}} W\,(V_u - V_v), \tag{31}$$

where $W \in \mathbb{R}^{d\times d}$ is a learnable channel-mixing matrix. The update (31) can be written compactly as

$$V \;\mapsto\; V - (LV)\,W^\top, \tag{32}$$

up to conventions on left/right multiplication. Thus, GT transport contains (as a special case) a Laplacian-driven smoothing step: it penalizes high variation across adjacent token positions.

Our implemented GT-Full transport is nonlinear: each edge update is produced by an MLP acting on (V_u, V_v) (and optional relation features), and the total update sums messages on incident edges. Nonetheless, the Laplacian picture remains valid under local linearization. Writing the GT transport increment as $\Delta_G(V)$, the first-order Taylor approximation around a representation V takes the form

$$\Delta_G(V+\delta) \;\approx\; \Delta_G(V) + J_G(V)\,\delta, \tag{33}$$

and, for many message-passing parameterizations, $J_G(V)$ is well-approximated by a Laplacian-like operator composed with a learned channel map, yielding a *data-dependent diffusion*:

$$V \;\mapsto\; V - \eta\, L_{\text{eff}}(V)\, V, \tag{34}$$

where $L_{\text{eff}}(V)$ is a (representation-dependent) Laplacian or Laplacian-like operator and $\eta > 0$ is an effective step size.

Spectral view: GT as a learned low-pass filter

Let $L = U\Lambda U^\top$ be the eigendecomposition of the (symmetric) Laplacian. In the linear regime (32), a single transport step acts on each Laplacian eigenmode as a multiplicative spectral filter. Ignoring cross-channel mixing for clarity, the update can be viewed as

$$V \;\mapsto\; \big(I - \eta L\big)\, V \quad\Longrightarrow\quad \widehat{V}_k \;\mapsto\; (1-\eta\lambda_k)\, \widehat{V}_k, \tag{35}$$

where $\widehat{V}_k$ denotes the projection of V onto eigenvector u_k. Since λ_k increases with frequency on the graph, high-frequency modes are damped more strongly than low-frequency modes. Repeating transport for multiple GT depth steps yields a polynomial (or, in the small-step limit, exponential) filter:

$$V \;\mapsto\; g(L)\, V, \qquad g(\lambda) \approx (1-\eta\lambda)^m \approx e^{-\eta m \lambda}. \tag{36}$$

This is precisely the heat-kernel/harmonic-analysis perspective: GT transport performs diffusion on token geometry.

A SMOOTHNESS ENERGY. A standard measure of variation of embeddings on a graph is the Dirichlet energy

$$\mathcal{E}_L(V) \;=\; \operatorname{tr}(V^\top L V) \;=\; \frac{1}{2}\sum_{(u,v)\in\mathcal{E}} \|V_u - V_v\|_2^2. \tag{37}$$

Laplacian transport decreases $\mathcal{E}_L(V)$ for sufficiently small step sizes. Thus GT implements an explicit inductive bias toward representations that vary smoothly over token positions.

Harmonic analysis on attention geometry

The preceding discussion used a fixed position graph (e.g., an undirected chain) as in our GT-Full implementation. A more geometric viewpoint treats attention as inducing a data-dependent graph on tokens. Let $A(x) \in \mathbb{R}^{n\times n}$

denote an attention affinity matrix (averaged over heads and symmetrized), and define an attention-Laplacian

$$L_A(x) \; = \; D_A(x) - \frac{1}{2}\big(A(x) + A(x)^\top\big), \tag{38}$$

$$D_A(x)_{ii} = \sum_j \tfrac{1}{2}(A_{ij}(x) + A_{ji}(x)). \tag{39}$$

A transport step of the form $V \; \mapsto \; V - \eta L_A(x) \, V$ performs diffusion *along attention neighborhoods*, i.e., along the learned geometry of token interactions. This motivates interpreting GT as performing a type of harmonic analysis on attention geometry: transport suppresses high-frequency components on the attention graph while preserving low-frequency (global/semantic) structure. Although our current GT-Full uses a fixed positional graph, this attention-induced Laplacian view clarifies why GT transport interacts strongly with the topology of attention patterns measured by our persistent-homology probes.

Connection to commutator energy and dynamic compositionality

Recall that our Čech-style obstruction proxy measures order sensitivity among patch maps (attention A, feed-forward F, and a geometric operator G in GT-Full). In the residual-update form, patch maps can be written as $T_i(x) = x + \Delta_i(x)$, and we use the scale-invariant delta-commutator ratio

$$\mathrm{Comm}_\Delta(T_1, T_2; x) = \frac{\|\Delta_{12}(x) - \Delta_{21}(x)\|_2^2}{\|\Delta_{12}(x)\|_2^2 + \|\Delta_{21}(x)\|_2^2 + \varepsilon}.$$

When G behaves approximately as a diffusion operator (contractive on high-frequency modes), it suppresses precisely the components of representations that tend to induce brittle, order-sensitive interactions between sub-operators. This yields an intuitive explanation for the empirical reduction in obstruction: transport makes intermediate representations smoother over the token graph, reducing interference between attention and feed-forward updates.

Remark 3 (GT transport as stabilizing preconditioner)**.** *Viewed through the Laplacian lens, GT transport acts as a learned preconditioner or regularizer on the residual stream. It reduces high-frequency variation on token geometry (or attention geometry), thereby stabilizing the composition of local updates and lowering commutator-based obstruction proxies in data-limited regimes.*

A contractive-transport bound for commutator energy

We now formalize the intuition that diffusion-like transport suppresses order-sensitivity. For clarity we work with a simplified (but informative) model in which the GT transport update is approximately linear and contractive on the token graph. The result can be read as a *first-order* commutator bound that explains why GT tends to reduce the Čech-style obstruction proxy.

SETUP. Let $T_A(x) = x + \Delta_A(x)$ be a residual patch map (e.g., attention or feed-forward), and let the transport patch be

$$T_G(x) = x + \Delta_G(x), \qquad \Delta_G(x) = -\eta\, Lx, \tag{40}$$

where L is a symmetric graph Laplacian (or Laplacian-like operator) and $\eta > 0$ is a step size. In the spectral basis $L = U\Lambda U^\top$, the map T_G is a frequency-dependent shrinkage operator.

We measure order-sensitivity using the *delta-commutator*:

$$\Delta_{AG}(x) = T_A(T_G(x)) - x, \qquad \Delta_{GA}(x) = T_G(T_A(x)) - x,$$

and the corresponding delta-commutator ratio $\mathrm{Comm}_\Delta(T_A, T_G; x)$.

Lemma 3 (First-order commutator bound under Laplacian transport)**.** *Assume Δ_A is differentiable and locally L_A-Lipschitz at x, i.e. $\|J_A(x)\| \le L_A$ where $J_A(x) = \nabla_x \Delta_A(x)$. Let T_G be the Laplacian transport (40). Then, for sufficiently small η,*

$$\|\Delta_{AG}(x) - \Delta_{GA}(x)\| \;\le\; \eta\,\|L\|\,\|\Delta_A(x)\| \;+\; \eta\, L_A\,\|Lx\| \;+\; O(\eta^2). \tag{41}$$

In particular, if x is predominantly low-frequency on the graph (so $\|Lx\|$ is small) and the residual update $\Delta_A(x)$ has bounded magnitude, then the order-sensitivity between A and G is $O(\eta)$.

Proof sketch. Write

$$T_A(T_G(x)) = x + \Delta_G(x) + \Delta_A(x + \Delta_G(x)), \tag{42}$$
$$T_G(T_A(x)) = x + \Delta_A(x) + \Delta_G(x + \Delta_A(x)). \tag{43}$$

Subtracting and using $\Delta_G(x) = -\eta Lx$ gives

$$\Delta_{AG}(x) - \Delta_{GA}(x) = \Delta_A(x - \eta Lx) - \Delta_A(x) \;+\; \eta L \Delta_A(x).$$

A first-order Taylor expansion yields $\Delta_A(x - \eta Lx) - \Delta_A(x) = -\eta\, J_A(x)\, Lx + O(\eta^2)$. Taking norms and using $\|J_A(x)\| \le L_A$ gives

$$\|\Delta_{AG}(x) - \Delta_{GA}(x)\| \le \eta\, L_A\, \|Lx\| + \eta\, \|L\|\, \|\Delta_A(x)\| + O(\eta^2),$$

which is (41). □

INTERPRETATION. Lemma 3 shows that Laplacian transport reduces order-sensitivity whenever the residual stream is smooth on the token graph (small $\|Lx\|$) and the transport step size η is modest. In GT-Full, the transport operator is nonlinear and data-dependent, but local linearization reduces to the same qualitative form: high-frequency (large-λ) components are damped more strongly than low-frequency ones, and the commutator magnitude is controlled by the residual high-frequency energy.

Measuring Dirichlet energy as a spectral regularization diagnostic

The Laplacian viewpoint suggests a simple, cheap-to-compute diagnostic that can be logged during training: the graph Dirichlet energy of the residual stream,

$$\mathcal{E}_L(V) = \operatorname{tr}(V^\top L V) = \frac{1}{2} \sum_{(u,v)\in\mathcal{E}} \|V_u - V_v\|_2^2.$$

For GT-Full with a fixed positional chain graph, L is fixed and $\mathcal{E}_L$ measures how rapidly embeddings vary across adjacent token positions. The theory predicts that (i) GT transport should reduce $\mathcal{E}_L$ relative to a baseline Transformer, and (ii) reductions in $\mathcal{E}_L$ should correlate with lower commutator-based obstruction, since $\|Lx\|$ controls the leading term in Lemma 3.

PRACTICAL LOGGING RECIPE. Given a batch embedding tensor $V \in \mathbb{R}^{B\times n\times d}$ and a fixed edge list $\mathcal{E}$ on positions, a minibatch estimate of Dirichlet energy is

$$\widehat{\mathcal{E}}_L(V) = \frac{1}{2B} \sum_{b=1}^{B} \sum_{(u,v)\in\mathcal{E}} \|V_{b,u} - V_{b,v}\|_2^2.$$

This can be computed efficiently by gathering endpoint embeddings for all edges and averaging squared differences. One may also normalize by nd to make values comparable across widths and sequence lengths.

PREDICTION. In data-limited regimes, we expect GT variants to exhibit (a) lower Dirichlet energy on the token graph and (b) lower delta-commutator obstruction, relative to baseline Transformers. In scaling regimes (increasing depth or context length), we expect GT to stabilize training by preventing uncontrolled growth of high-frequency energy and hence mitigating commutator-energy drift.

Implications for scaling and sample efficiency

The Laplacian/harmonic-analysis perspective suggests two testable predictions: (i) GT variants should be more sample efficient than baseline Transformers in settings where data are limited relative to model capacity, since transport acts as an explicit smoothness prior; and (ii) GT should expand stable depth/context scaling regimes by damping high-frequency instability modes that otherwise amplify across compositions. These predictions align with our observed separation between GT and Pre/Post-LN Transformers in low-data seq2seq regimes, and motivate deeper scaling studies on language modeling benchmarks.

Data-Limited WikiText-103: GT Improves Sample Efficiency

To probe sample efficiency and learning-time compositionality in a constrained setting (motivated by limited on-device compute), we evaluate

Model	Val CE ↓	Val PPL ↓	Tok Acc ↑	Cech ↓	Dir E_{end} ↓	H_1 ↓	Bars
GT_Full_PreLN_L4	**6.536**	**689.8**	0.181	2.84e-02	**0.873**	1.88e-02	6.500
GT_MoE_PreLN_L4	6.538	690.7	**0.182**	5.54e-02	0.924	3.33e-02	11.000
GT_Lite_PreLN_L4	6.557	704.4	0.181	**2.68e-02**	0.875	7.44e-03	3.000
GT_Full_PostLNRes_L4	6.611	743.2	0.179	4.73e-02	0.993	1.68e-02	5.000
GT_Lite_PostLNRes_L4	6.630	757.6	0.174	4.54e-02	0.937	4.07e-02	10.000
GT_MoE_PostLNRes_L4	6.697	810.3	0.176	4.85e-02	1.085	7.95e-04	1.500
TF_PreLN_L4	8.976	7908	0.035	4.49e-02	1.009	7.54e-03	4.000
TF_PostLN_L4	9.105	9000	0.036	5.74e-02	1.053	1.77e-03	1.500

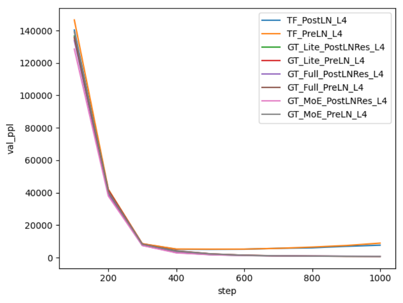
Table 4: Data-limited WikiText-103 (n_train=1000, n_val=100, seq_len=64, depth $L = 4$, 1000 steps). We report final validation cross-entropy (CE), perplexity (PPL), token accuracy, the Čech obstruction proxy (delta-commutator ratio), normalized chain Dirichlet energy at the encoder output, and attention-geometry topology probes (total H_1 persistence and number of H_1 bars).

baseline Transformers and GT variants on a *data-limited* WikiText-103 prefix→suffix seq2seq task. We subsample $n_{\text{train}} = 1000$ sentences and $n_{\text{val}} = 100$ sentences, clip sentences to length $L \leq 64$, and train for 1000 iterations with evaluation every 100 steps. All models use the same hidden size ($d_{\text{model}} = 96$), heads (4), depth (4 encoder/decoder blocks), optimizer, and learning-rate schedule.

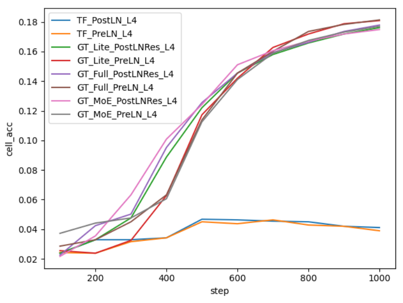
MAIN FINDING: STRONG SEPARATION UNDER LIMITED DATA. The figure shows that GT variants achieve markedly better validation loss/perplexity and token accuracy than both Post-LN and Pre-LN Transformers within the same training budget. At step 1000, GT models reach token accuracy $\approx$ 0.17–0.18 while baseline Transformers remain near $\approx$ 0.04 (a $\sim 4\times$ gap), and validation cross-entropy improves from $\approx$ 9 (Transformers) to $\approx$ 6.6–6.7 (GT), corresponding to a large perplexity reduction (since $\text{ppl} = \exp(\text{CE})$). Sequence-level accuracy remains near zero for all models, as exact suffix matching is an extremely strict metric for natural language.

SUMMARY TABLE. The table summarizes final-step performance and instrumental probes for all model variants. GT-Full with Pre-LN achieves the best validation cross-entropy (6.536) and among the best token accuracy (0.181), reducing perplexity by $\sim 11.5\times$ relative to the best Transformer baseline (TF-PreLN, PPL $\approx$ 7908). Within each GT family, Pre-LN systematically improves sample efficiency and reduces the commutator-based obstruction proxy. Moreover, GT transport reduces the chain-graph Dirichlet energy of encoder representations, consistent with the interpretation of GT as a learned diffusion regularizer on token geometry.

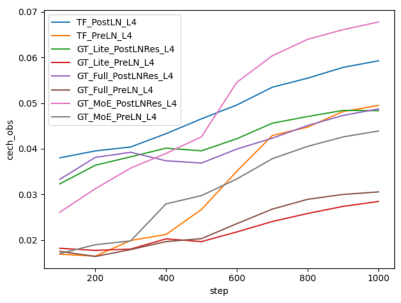
ČECH OBSTRUCTION DISTINGUISHES ARCHITECTURES. The figure also plots the Čech-style obstruction proxy (delta-commutator ratio) computed on a fixed probe batch. In this data-limited regime, GT-Lite/GT-Full with Pre-LN exhibit substantially lower obstruction than the baseline Transformers and lower than their PostLNResidual counterparts, consistent with the view that normalization placement and geometric transport jointly reduce order-sensitive interference between patch updates during learning.



(a) Validation perplexity



(b) Token accuracy

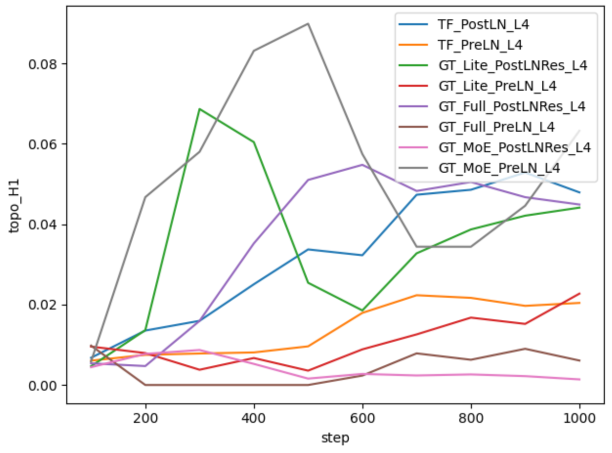


(c) Čech obstruction proxy

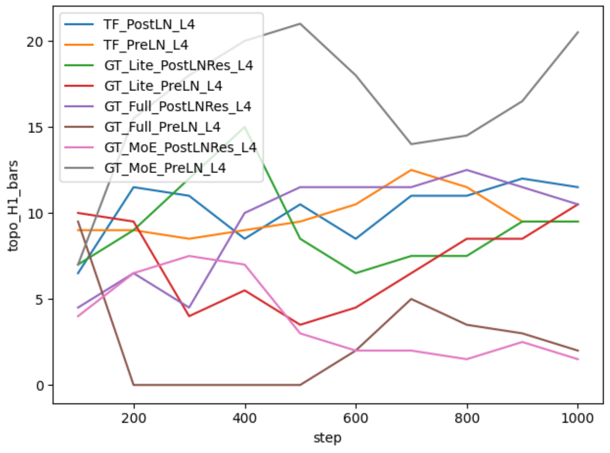
Figure 19: **Data-limited WikiText-**

),

n



(a) H_1 total persistence



r

than baseline Transformers on a fixed probe batch.

(b) H_1 bar count

Figure 20: **Attention-geometry diagnostics in the data-limited WikiText-103 regime.** Persistent-homology summaries of the (symmetrized) encoder attention affinity reveal architecture-dependent differences in the complexity of learned attention geometry, complementing commutator-based obstruction measures.

ATTENTION-GEOMETRY DIAGNOSTICS. In addition to commutator-based obstruction, we log persistent-homology summaries of the encoder attention graph (total H_1 persistence and number of H_1 bars). These topology probes expose qualitative differences in the evolving attention geometry across architectures, complementing the obstruction proxy by measuring global geometric structure rather than local operator compatibility. We include the corresponding plots are shown in the below figures.

Data-Limited WikiText-2: similar separation

We repeated the same data-limited protocol on WikiText-2 (prefix→suffix seq2seq; $n_{\text{train}} = 1000$, $n_{\text{val}} = 100$, sentence length clipped to $L \leq 64$, 1000 optimization steps, evaluation every 100 steps) and observe the same qualitative separation: GT variants achieve dramatically better validation cross-entropy/perplexity and token accuracy than both Post-LN and Pre-LN Transformers under the same compute budget.

SUMMARY. The table reports final-step metrics and instrumental probes. The best Transformer baseline (TF-PreLN) achieves validation CE 7.903 (PPL $\approx$ 2705) and token accuracy 0.041, whereas the best GT variant (GT-

Model	Val CE ↓	Val PPL ↓	Tok Acc ↑	Cech ↓	Dir E_{end} ↓	H_1 ↓	Bars
TF_PostLN_L4	7.979	2918	0.035	5.20e-02	1.253	1.31e-01	9.0
TF_PreLN_L4	7.903	2705	0.041	3.97e-02	1.252	3.43e-02	8.5
GT_Lite_PostLNRes_L4	4.884	132.2	0.240	4.75e-02	1.185	8.38e-02	8.0
GT_Lite_PreLN_L4	**4.766**	**117.5**	**0.245**	**2.97e-02**	**1.023**	8.04e-02	14.5
GT_Full_PostLNRes_L4	4.903	134.7	0.241	4.55e-02	1.164	4.59e-02	9.5
GT_Full_PreLN_L4	4.862	129.3	0.242	3.09e-02	1.054	7.40e-02	6.5
GT_MoE_PostLNRes_L4	5.020	151.4	0.236	5.18e-02	1.317	1.08e-01	16.0
GT_MoE_PreLN_L4	4.903	134.7	0.242	4.92e-02	1.083	2.29e-02	7.0

Table 5: Data-limited WikiText-2 (n_train=1000, n_val=100, seq_len=64, depth $L = 4$, 1000 steps). We report final validation cross-entropy (CE), perplexity (PPL), token accuracy, the Čech obstruction proxy (delta-commutator ratio), normalized chain Dirichlet energy at the encoder output, and attention-geometry topology probes (total H_1 persistence and number of H_1 bars).

Lite PreLN) achieves validation CE 4.766 (PPL $\approx$ 117.5) and token accuracy 0.245. This corresponds to a reduction of 3.14 nats in CE (a $\sim 23\times$ perplexity improvement) and a $\sim 5.9\times$ gain in token accuracy in the same training budget. Within each GT family, Pre-LN improves performance and typically reduces the commutator-based obstruction proxy and the chain Dirichlet energy at the encoder output, consistent with GT acting as a learned diffusion regularizer on token geometry.

Implementation details and reproducibility. All models are trained with the same optimization loop and logging harness used throughout this work. We use AdamW with base learning rate 3×10^{-4}, linear warmup for the first 500 steps, and global-norm gradient clipping at 1.0. We evaluate every 100 steps and compute the Čech obstruction proxy, Dirichlet energy, and attention-geometry (persistent-homology) probes at the same cadence on a *fixed probe batch* drawn once from the training-evaluation loader (20 cached minibatches) to ensure comparability across steps and across model families. For the obstruction proxy we use the scale-invariant delta-commutator ratio; for Dirichlet energy we compute the normalized chain-graph Dirichlet energy of the encoder residual stream at the encoder output]; and for topology we compute persistent homology on the symmetrized, head-averaged self-attention affinity restricted to the first 16 tokens and the first 2 encoder layers to keep PH overhead modest. The WikiText-103 data-limited setting uses a prefix$\rightarrow$ suffix seq2seq construction: the raw token stream is split into sentences at `<eos>`, sentences are clipped to length 64, and for each sentence a random pivot selects a prefix as encoder input and a suffix as decoder target with teacher forcing (decoder input is `[BOS, y_0, ..., y_t-2]`). Padding uses token id 0 and is ignored in the cross-entropy loss. All results in the table and the figures were produced with the command:

```
python seq2seq_scaling_PrePostLN_compare.py -task wiki103
                        -seq_len 64
 -num_iters 1000 -n_train 1000 -n_val 100 -steps_per_eval
                          100.
```

We fix the random seed at 0 for the plots shown; in ongoing work we sweep multiple seeds and larger training budgets on dedicated hardware.

Sheaf Laplacians as a principled match to Čech obstruction

We interpreted GT transport through the lens of the (scalar) graph Laplacian, yielding a diffusion/harmonic-analysis view on token geometry. While useful, the standard Laplacian treats features as scalars or as independently smoothed channels. A closer mathematical match to our Čech-style obstruction proxy is provided by *sheaf Laplacians*, which explicitly model compatibility constraints and learned restriction/transport maps on overlaps. This connects GT more directly to Čech (co)homology and to holonomy/curvature-style inconsistency measures.

A TOKEN-POSITION SHEAF. Let $G = (V, E)$ be a graph on token positions (e.g., a chain). A (cellular) sheaf $\mathcal{F}$ assigns a vector space (a *stalk*) to each vertex $v \in V$ and to each edge $e \in E$, together with linear restriction maps from vertices to incident edges. For our setting, take

$$\mathcal{F}(v) = \mathbb{R}^d \quad (v \in V), \qquad \mathcal{F}(e) = \mathbb{R}^d \quad (e \in E),$$

and for each edge $e = (u, v)$ define restriction maps

$$\rho_{u \to e} : \mathbb{R}^d \to \mathbb{R}^d, \qquad \rho_{v \to e} : \mathbb{R}^d \to \mathbb{R}^d.$$

In the *constant sheaf* case, $\rho_{u \to e} = \rho_{v \to e} = I$, and the resulting sheaf Laplacian reduces to a standard graph Laplacian acting on each feature channel.

LEARNED RESTRICTIONS AS TRANSPORT/CONNECTION. To connect to GT, we interpret the edge-wise GT mechanism as implicitly learning compatibility maps on overlaps. Concretely, the edge MLP in simplicial message passing produces a message depending on (V_u, V_v) (and relation features), which can be linearized locally as a learned transformation aligning feature coordinates across an edge. This suggests parameterizing $\rho_{u \to e}$ and $\rho_{v \to e}$ as *learned transport maps* (e.g., orthogonal or low-rank maps) that map vertex embeddings into a shared edge coordinate system.

COBOUNDARY AND SHEAF LAPLACIAN. Let $C^0 = \bigoplus_{v \in V} \mathcal{F}(v) \cong \mathbb{R}^{|V|d}$ be the space of 0-cochains (assignments of a d-vector to each token position), and $C^1 = \bigoplus_{e \in E} \mathcal{F}(e) \cong \mathbb{R}^{|E|d}$ the space of 1-cochains. The sheaf coboundary operator $\delta : C^0 \to C^1$ is defined edgewise by

$$(\delta x)_e = \rho_{v \to e}(x_v) - \rho_{u \to e}(x_u), \qquad e = (u, v) \in E, \tag{44}$$

which measures inconsistency of the two endpoint assignments after mapping into the edge stalk. The *sheaf* 0*-Laplacian* is then

$$\Delta_{\mathcal{F}}^{(0)} = \delta^\top \delta, \tag{45}$$

with respect to the natural inner products on C^0 and C^1 (or weighted variants). In the constant-sheaf case this reduces (up to scaling conventions) to the standard graph Laplacian; in the orthogonal-transport case it yields a connection-Laplacian-type operator whose spectrum encodes the consistency of transport around cycles.

A SHEAF DIRICHLET ENERGY AS A ČECH-STYLE OBSTRUCTION. Equation (44) provides a direct, cohomology-native measurement of "failure to glue":

$$\mathcal{E}_{\text{sheaf}}(x) \;=\; \frac{\|\delta x\|_2^2}{\|x\|_2^2+\varepsilon} \;=\; \frac{x^\top \Delta_{\mathcal{F}}^{(0)} x}{\|x\|_2^2+\varepsilon}. \tag{46}$$

This *sheaf Dirichlet energy* is a principled analogue of our Čech-style obstruction proxy: rather than measuring order sensitivity of patch-map composition via commutators, it measures incompatibility of local assignments through the sheaf coboundary itself. In particular, $\mathcal{E}_{\text{sheaf}}(x)$ is small when vertex features admit a globally consistent "gluing" under the learned restrictions, and large when local overlap constraints are violated.

HODGE-THEORETIC INTERPRETATION. Sheaf Laplacians support a Hodge decomposition: the nullspace $\ker(\Delta_{\mathcal{F}}^{(0)})$ corresponds to globally consistent 0-cochains (harmonic sections), while the spectrum of $\Delta_{\mathcal{F}}^{(0)}$ quantifies the cost of inconsistency at different modes. When the restriction maps encode a non-flat connection, inconsistencies accumulate around loops; this is reflected spectrally (e.g., through elevated low-frequency eigenvalues) and aligns with the intuition behind our commutator-based loop inconsistency probes.

RELATION TO GT MECHANISMS. This perspective suggests that GT transport can be viewed as learning (explicitly or implicitly) a sheaf/connection over token positions, with transport maps that reduce coboundary energy and thereby suppress learning-time compositionality obstructions. It also suggests a natural extension of GT: construct a data-dependent sheaf using attention-induced graphs, then apply a learned sheaf Laplacian to perform harmonic analysis *on attention geometry* rather than on a fixed positional graph. We leave a full empirical evaluation of sheaf Dirichlet energy and sheaf-Laplacian spectra to future work, but note that this framework provides a mathematically direct bridge between GT transport, Čech cohomology, and our obstruction measurements.

PRACTICAL LOGGING RECIPE: SHEAF DIRICHLET ENERGY ON A TOKEN GRAPH. The sheaf Dirichlet energy (46) can be logged during training with overhead comparable to the (scalar) Dirichlet energy, provided one specifies restriction maps $\rho_{u\to e}, \rho_{v\to e}$ for each edge $e = (u, v)$. For a minibatch tensor $X \in \mathbb{R}^{B\times n\times d}$ representing encoder residual streams on n token positions, define the edgewise coboundary residual for each sample b

as

$$(\delta X^{(b)})_e \;=\; \rho_{v\to e}\big(X_v^{(b)}\big) - \rho_{u\to e}\big(X_u^{(b)}\big), \qquad e=(u,v)\in E.$$

A normalized minibatch estimate is then

$$\widehat{\mathcal{E}}_{\text{sheaf}}(X) = \frac{1}{B|E|d}\sum_{b=1}^{B}\sum_{e\in E}\|(\delta X^{(b)})_e\|_2^2, \tag{47}$$

which is directly comparable across sequence lengths and widths.

Choosing restriction maps. Several choices are natural and correspond to increasingly structured notions of "gluing":

- **Constant sheaf (baseline):** $\rho_{u\to e}=\rho_{v\to e}=I$, in which case (47) reduces (up to constants) to the usual chain Dirichlet energy and measures feature smoothness across adjacent tokens.
- **Connection/orthogonal transport:** enforce $\rho_{u\to e},\rho_{v\to e}\in O(d)$ (e.g., parameterize each as an orthogonal matrix, or as a product of a small number of Householder reflections). This measures inconsistency after aligning local coordinate frames and yields a connection-Laplacian-type operator.
- **GT-induced linearization:** define $\rho_{u\to e}$ and $\rho_{v\to e}$ from the local linearization of the GT edge MLP at the current representation (e.g., use Jacobians of the message map with respect to X_u and X_v). This yields a data-dependent sheaf that is closest to the actual GT mechanism, at the cost of higher computation.

In practice, the first two options already provide useful diagnostics: the constant-sheaf energy captures raw smoothness, while the orthogonal-transport energy captures "smoothness up to learned alignment" and is a closer analogue of Čech-style gluing.

Relationship to the commutator obstruction proxy. Both the sheaf Dirichlet energy and the commutator-based Čech obstruction proxy measure failure to glue local information globally, but at different granularities. The coboundary energy (47) measures overlap inconsistency edge-by-edge (a linearized, cohomology-native notion), whereas the commutator proxy measures loop inconsistency between *patch maps* (a nonlinear, composition-sensitive notion). Empirically, we expect these metrics to correlate when transport is diffusion-like (so that high-frequency edge inconsistencies drive loop inconsistencies), while diverging in settings where routing or nonlinear normalization induces order sensitivity beyond what edgewise alignment captures. Logging both quantities provides a direct way to disentangle "geometric roughness" from "operator-order interference" during learning.

Discussion and Predictions

This work positions commutator energy as a unifying order parameter for analyzing learning-time compositionality in deep neural networks. By connecting empirical stability phenomena in Transformers to mean-field analysis of Jacobian non-commutativity, we provide a principled explanation for the success of Pre-LN architectures and for the further gains achieved by Geometric Transformers (GT). We conclude by summarizing the implications of this perspective and outlining testable predictions suggested by the theory.

Commutator Energy as a Diagnostic of Learning Dynamics

Traditional analyses of deep networks focus on gradient norms, signal variance, or spectral properties of individual layers. Commutator energy captures a complementary phenomenon: the compatibility of learned sub-operator deformations when composed dynamically during training. High commutator energy indicates that small changes in one sub-operator significantly alter the effective behavior of others, leading to gradient interference and instability. Low commutator energy indicates approximate exchangeability of updates and more predictable learning dynamics.

This distinction explains why commutator energy often continues to evolve even after task loss has saturated, as observed in residual MLPs, sequence-to-sequence models, and large-scale language modeling. The probe therefore measures an aspect of learning dynamics that is largely invisible to standard task metrics.

Normalization, Alignment, and Architectural Design

Our analysis clarifies the respective roles of normalization and geometric alignment in stabilizing deep networks. Layer normalization primarily controls the *scale* of Jacobians and suppresses variance-driven instability. This accounts for the empirical success of Pre-LN Transformers and related normalization-based fixes.

Geometric alignment mechanisms, as implemented in GT architectures, operate at a different level. By coordinating the *directional structure* of learned transformations, alignment suppresses Jacobian non-commutativity directly. This suggests that normalization and alignment are complementary rather than competing design principles: normalization stabilizes magnitudes, while alignment stabilizes composition.

Predictions

The mean-field commutator-energy framework yields several concrete predictions that are testable empirically:

DEPTH SCALING. For architectures lacking explicit alignment mechanisms, expected commutator energy should grow with depth, even when gradients remain well-scaled. Pre-LN normalization reduces the growth rate but does not eliminate it. Architectures with explicit geometric alignment should exhibit bounded or slowly growing commutator energy with depth.

WIDTH SCALING. In the infinite-width limit, commutator energy at initialization converges to a nonzero constant for generic residual architectures. Normalization reduces this constant multiplicatively, while alignment mechanisms reduce it further by altering cross-covariance structure between Jacobians.

TASK INDEPENDENCE. Because commutator energy arises from operator interactions rather than task loss, its qualitative behavior should be largely task-independent. Architectures that suppress commutator energy in simple settings (e.g., residual MLPs on two-moons) should do so consistently in large-scale sequence modeling and other domains.

CONDITIONAL COMPUTATION. Architectures that rely on state-dependent branching (e.g., mixture-of-experts) should exhibit intermediate commutator-energy behavior: lower than fully uncoordinated baselines, but higher and more variable than explicitly aligned architectures. This prediction aligns with observed GT-MoE behavior.

Implications for Optimization and Generalization

Although commutator energy is defined purely in terms of forward computations, its evolution correlates strongly with optimization stability and generalization. This suggests that architectural control of learning-time geometry—rather than loss shaping or optimizer tuning alone—is a critical factor in scaling deep models. From this perspective, Diagrammatic Backpropagation can be viewed as an explicit mechanism for penalizing destructive order sensitivity, while GT architectures provide structural support that makes such control feasible.

Limitations and Open Questions

Our mean-field analysis focuses on initialization and early training regimes and does not capture all phenomena observed in finite-width or late-stage training. A complete theory of commutator energy during training would require coupling mean-field analysis with non-equilibrium dynamics and representation drift. Additionally, extending the analysis to attention-specific Jacobians and token-dependent routing remains an open challenge.

Nevertheless, the qualitative alignment between theory and empirical results across architectures and tasks suggests that commutator energy captures a fundamental aspect of learning-time compositionality.

Summary and Further Reading

We proposed a mean-field theory of Geometric Transformers. There is substantial related work to explore on the general mean-field theory of deep learning. Mean-field theory has played a central role in understanding signal propagation, gradient dynamics, and stability in deep neural networks. Early work analyzed variance propagation and critical initialization in wide networks [56].

[56] Ben Poole, Subhaneil Lahiri, Maithra Raghu, Jascha Sohl-Dickstein, and Surya Ganguli. Exponential expressivity in deep neural networks through transient chaos. *Advances in Neural Information Processing Systems*, 29, 2016. URL https://arxiv.org/abs/1606.05340; and Samuel S. Schoenholz, Justin Gilmer, Surya Ganguli, and Jascha Sohl-Dickstein. Deep information propagation. *International Conference on Learning Representations*, 2017. URL https://arxiv.org/abs/1611.01232

Scaling Laws for Geometric Transformers

Transformer architectures exhibit remarkable empirical scalability, yet their training dynamics remain poorly understood, particularly as depth increases. Recent work has shown that learning instabilities in deep Transformers correlate with the accumulation of order-sensitivity between residual sub-operators, which can be quantified by a forward-pass diagnostic termed commutator energy. In this chapter, we study how commutator energy scales with depth in standard Post-LN and Pre-LN Transformers, as well as in Geometric Transformer (GT) architectures that explicitly control learning-time geometry.

We show empirically that Pre-LN normalization suppresses the magnitude of commutator energy as depth increases, but does not prevent generalization collapse, revealing a previously unrecognized regime in which normalization alone is insufficient. In contrast, Geometric Transformers—particularly GT-Full variants with explicit geometric transport—exhibit depth-stable commutator energy and sustained generalization performance. We interpret these results through a mean-field and Jacobian-based analysis, arguing that GT architectures alter the scaling law itself by enforcing directional alignment of learned transformations rather than merely suppressing variance.

Our findings suggest that geometric alignment constitutes a distinct scaling mechanism complementary to width and data scaling, and provide a new theoretical lens for understanding why deep Transformers require increasingly elaborate stabilization strategies at scale.

Introduction

Transformer architectures have demonstrated extraordinary empirical scalability, underpinning modern advances in language modeling, vision, and multimodal learning. Over the past several years, a substantial body of work has established that model performance improves predictably as a function of scale—typically measured in parameters, data, or compute—leading to the formulation of *scaling laws* that guide practical system design. Despite this success, the internal learning dynamics of deep Transformers remain poorly understood, particularly as depth increases and residual interactions accumulate.

A growing line of evidence suggests that training instability in deep Transformers is not merely an optimization artifact, but a consequence of how residual updates interact geometrically on the representation manifold. Standard Transformer blocks apply multiple nonlinear sub-operators—most notably self-attention and feed-forward transformations—sequentially to a shared residual stream. While the forward computation graph is statically compositional, the *learning dynamics* induced by gradient descent need not respect this compositional structure. As depth increases, incompatible deformations can accumulate, leading to generalization collapse even when training loss continues to decrease.

Recent architectural advances such as Pre-LayerNorm (Pre-LN) Transformers have substantially improved optimization stability at depth by relocating normalization before residual updates. Mean-field analyses have shown that Pre-LN controls gradient magnitudes and suppresses exploding or vanishing signals, enabling the successful training of deeper networks. However, Pre-LN does not fully explain why deep Transformers often continue to exhibit poor generalization behavior, nor why increasing depth can sometimes worsen validation performance despite improved optimization metrics.

In previous chapters, we introduced *Diagrammatic Backpropagation* (DB) and the *Geometric Transformer* (GT) as a framework for diagnosing and controlling learning-time inconsistencies in compositional neural architectures. Central to this framework is a forward-pass diagnostic termed *commutator energy*, which measures order sensitivity between learned sub-operators within a residual block. Empirically, we showed that standard Transformers accumulate commutator energy during training, while GT variants systematically suppress it through explicit geometric alignment mechanisms.

In this chapter, we elevate commutator energy from an architectural diagnostic to a *scaling observable*. Our central thesis is that depth scaling in Transformers induces distinct dynamical regimes that cannot be explained by normalization alone. Specifically, we show that:

1. Pre-LN Transformers exhibit a depth-dependent suppression of commutator magnitude, yet still enter regimes of poor generalization as depth increases.

2. Geometric Transformers alter the depth-scaling behavior itself, maintaining bounded commutator energy and stable generalization across increasing depth.

3. Normalization and geometric alignment control fundamentally different aspects of learning dynamics: normalization limits variance, while geometric alignment constrains the relative orientation of learned transformations.

Through controlled scaling experiments on synthetic sequence-to-sequence tasks and truncated WikiText-103 language modeling, we demonstrate that commutator energy exhibits characteristic scaling behavior as depth increases. Pre-LN Transformers transition into a regime of *suppressed but misaligned* updates, whereas GT architectures remain in a regime of *dynamically compositional* learning even at greater depth.

We provide a Jacobian- and mean-field-based interpretation of these findings, arguing that GT architectures enforce directional alignment between residual sub-operators rather than merely controlling their magnitude. This perspective naturally generalizes existing mean-field analyses of Transformers and suggests a new class of *geometry-aware scaling laws* in which depth, alignment, and dynamic compositionality jointly determine learning behavior.

More broadly, our results suggest that geometric structure is not merely a representational convenience, but a *scaling mechanism*. By modifying how residual updates compose as depth increases, Geometric Transformers shift the qualitative behavior of deep learning systems into a more stable and predictable regime. This work thus complements classical scaling laws by introducing geometric alignment as a first-class variable in the theory of deep neural scaling.

Problem Setup and Scaling Observables

This paper studies the depth-scaling behavior of Transformer and Geometric Transformer architectures through the lens of learning-time compositionality. Our goal is not to propose a new task or optimization method, but to identify *scaling observables* that characterize how internal learning dynamics change as model depth increases.

We focus on encoder–decoder sequence models trained with standard gradient-based optimization under identical data, parameter, and training-budget constraints, varying only architectural structure and depth. The central object of study is a forward-pass diagnostic—*commutator energy*—which measures order sensitivity between learned sub-operators within residual blocks.

Architectural Setting

We consider a family of models parameterized by depth L, hidden dimension d, and feed-forward expansion factor γ, trained on sequence-to-sequence language modeling tasks. All models share the same token embeddings, positional encodings, optimizer, batch size, and learning-rate schedule unless otherwise specified.

BASELINE TRANSFORMER. A standard Transformer encoder block consists of two residual sublayers applied sequentially to a shared residual stream:

1. multi-head self-attention,

2. position-wise feed-forward transformation.

Each sublayer is wrapped with residual addition and layer normalization. We consider both Post-LayerNorm (Post-LN) and Pre-LayerNorm (Pre-LN) variants.

GEOMETRIC TRANSFORMERS. Geometric Transformer (GT) variants augment the baseline block with an additional operator that alters the geometry of the residual stream:

- **GT-Lite**: inserts a local geometric smoothing operator (e.g., convolution or neighborhood mixing);
- **GT-Full**: inserts an explicit geometric transport operator based on message passing over a fixed relational or positional graph;
- **GT-MoE**: replaces the feed-forward stage with a mixture-of-experts operator with state-dependent routing.

All variants preserve the external Transformer interface and parameter order of magnitude.

Residual Sub-Operators and Effective Updates

We abstract each encoder block as a composition of *effective sub-operators* acting on the same representation. Let $x \in \mathbb{R}^{B\times T\times d}$ be the input to a block. Each sublayer defines an update of the form

$$T_i(x) \; = \; x + \Delta_i(x),$$

where $\Delta_i(x)$ includes the nonlinear transformation, residual addition, and normalization applied by the sublayer.

In a baseline Transformer block, we identify two such operators:

- A: the attention operator,
- F: the feed-forward operator.

In GT variants, an additional operator C (GT-Lite), G (GT-Full), or M (GT-MoE) is present.

Although the forward pass applies these operators in a fixed order, learning dynamics depend on how these updates interact on the representation manifold visited during training.

Commutator Energy as an Order-Sensitivity Measure

To quantify interaction effects between sub-operators, we define the *commutator energy* between two operators T_i and T_j evaluated at a representation x:

$$E_{\text{comm}}(T_i, T_j; x) \; = \; \frac{\|T_i(T_j(x)) - T_j(T_i(x))\|_2^2}{\|x\|_2^2 + \varepsilon},$$

where $\varepsilon > 0$ is a small constant for numerical stability.

This quantity measures how sensitive the effective update is to the order in which two sub-operators are applied. Importantly, we do *not* require T_i and T_j to commute as functions. Rather, large commutator energy indicates that the learned deformations induced by the two sub-operators are incompatible on the current representation manifold.

LAYERWISE AND MODEL-LEVEL OBSERVABLES. For a block containing multiple sub-operators, we compute the average commutator energy over all relevant operator pairs. For a model with L encoder layers, we define the model-level commutator energy as the average over layers:

$$E_{\text{comm}}^{\text{model}} = \frac{1}{L}\sum_{\ell=1}^{L} E_{\text{comm}}^{(\ell)}.$$

All measurements are performed in evaluation mode using forward passes only, without gradient computation.

Scaling Variables

We treat commutator energy as a *scaling observable* and study its dependence on the following variables:

- **Depth** L: number of encoder layers;
- **Architecture**: Transformer, GT-Lite, GT-Full, or GT-MoE;
- **Normalization regime**: Post-LN vs. Pre-LN;
- **Training time**: number of optimization steps.

Throughout the paper, width d, dataset size, and optimization hyperparameters are held fixed unless explicitly stated. This isolates the effect of depth and architectural geometry on learning dynamics.

Interpretation as a Learning-Time Order Parameter

Commutator energy plays the role of an *order parameter* for dynamic compositionality. Low commutator energy corresponds to approximate exchangeability of learned updates: the effect of applying sub-operators does not depend strongly on order. High commutator energy indicates destructive interference between updates, leading to unstable learning and poor generalization.

Crucially, commutator energy is computed entirely from forward evaluations and does not alter training unless explicitly added as a loss term. It therefore serves as an unbiased diagnostic of internal learning dynamics.

CONNECTION TO SCALING. As depth increases, residual updates accumulate across layers. Whether this accumulation remains stable depends on how sub-operator interactions scale with depth. In the following sections, we show that different architectures induce distinct scaling regimes for commutator energy, revealing qualitative differences in how deep networks compose learned transformations.

SUMMARY. This section has established the experimental and conceptual framework for our scaling analysis. We model Transformer blocks as interacting residual sub-operators, introduce commutator energy as a forward-pass measure of order sensitivity, and treat depth as a scaling variable. The remaining sections analyze how normalization and geometric alignment shape the depth-scaling behavior of this observable.

Initialization and Mean-Field Scaling of Commutator Energy

We begin our scaling analysis by studying the behavior of commutator energy at initialization. This provides a baseline prediction for how order sensitivity should scale with depth in the absence of learned geometric alignment and clarifies the role of normalization in early training dynamics.

Our analysis follows a mean-field perspective: we treat residual sub-operators as random transformations acting on high-dimensional representations and analyze the expected magnitude of their interaction under simplifying independence assumptions.

Residual Updates at Initialization

Consider two residual sub-operators T_i and T_j acting on a shared representation $x \in \mathbb{R}^d$:

$$T_i(x) = x + \Delta_i(x), \quad T_j(x) = x + \Delta_j(x),$$

where Δ_i and Δ_j are nonlinear functions parameterized by randomly initialized weights.

At initialization, we assume:

- weights are independently sampled with variance chosen to preserve activation scale;
- $\Delta_i(x)$ and $\Delta_j(x)$ are approximately mean-zero;
- Jacobians $J_i = \nabla\Delta_i(x)$ and $J_j = \nabla\Delta_j(x)$ behave as random matrices with bounded spectral norm.

Under these assumptions, the first-order expansion of the commutator residual yields

$$T_i(T_j(x)) - T_j(T_i(x)) \;\approx\; (J_iJ_j - J_jJ_i)\,x,$$

so commutator energy is controlled by the non-commutativity of the Jacobians.

Expected Commutator Energy at Initialization

Let $J_i, J_j \in \mathbb{R}^{d\times d}$ be independent random matrices drawn from a rotationally invariant ensemble with variance σ^2/d. A standard calculation shows that

$$\mathbb{E}\left[\|J_iJ_j - J_jJ_i\|_F^2\right] \;=\; \Theta(d\,\sigma^4),$$

even though $\mathbb{E}[J_iJ_j] = \mathbb{E}[J_jJ_i] = 0$.

Normalizing by $\|x\|_2^2 \sim d$, the expected commutator energy at initialization satisfies

$$\mathbb{E}\left[E_{\mathrm{comm}}(T_i, T_j; x)\right] \;=\; \Theta(\sigma^4),$$

a nonzero constant independent of width but sensitive to variance scaling.

This implies that even before training, residual sub-operators are generically *order-sensitive*. Commutator energy is therefore not a training artifact, but a structural property of deep residual architectures.

Depth Scaling Without Alignment

In a network of depth L, each encoder layer contributes its own commutator residual. Under independence assumptions, the expected model-level commutator energy scales as

$$\mathbb{E}\left[E_{\mathrm{comm}}^{\mathrm{model}}\right] \;\sim\; \frac{1}{L}\sum_{\ell=1}^{L}\mathbb{E}\left[E_{\mathrm{comm}}^{(\ell)}\right] \;\approx\; \Theta(\sigma^4),$$

remaining $O(1)$ per layer.

However, learning dynamics accumulate these order-sensitive interactions across layers. In the absence of corrective mechanisms, small incompatibilities between sub-operators compound with depth, leading to:

- increasing gradient interference,
- sensitivity to update ordering,
- degradation of generalization at fixed compute budgets.

This predicts a depth-dependent instability regime even when initialization is well-scaled.

Effect of Layer Normalization

Layer normalization alters this picture by rescaling inputs to each residual update. In Pre-LN architectures, normalization is applied *before* nonlinear transforms, effectively controlling the magnitude of Jacobians:

$$J_i^{\mathrm{PreLN}} \;\approx\; \alpha J_i, \quad \alpha < 1.$$

As a result,

$$\mathbb{E}\big[\|J_i^{\mathrm{PreLN}} J_j^{\mathrm{PreLN}} - J_j^{\mathrm{PreLN}} J_i^{\mathrm{PreLN}}\|_F^2\big] \;\sim\; \alpha^4\, \mathbb{E}\big[\|J_i J_j - J_j J_i\|_F^2\big].$$

Thus, Pre-LN reduces the *scale* of commutator energy at initialization, explaining its improved optimization stability relative to Post-LN Transformers.

Crucially, however, normalization does not alter the *directional structure* of J_i and J_j. Their eigenspaces remain unaligned, and commutator energy remains strictly positive.

Implications for Scaling

The mean-field analysis yields three immediate conclusions:

1. Commutator energy is generically nonzero at initialization in residual architectures.

2. Normalization reduces its magnitude but does not eliminate it.

3. Increasing depth exacerbates the accumulation of order sensitivity unless additional alignment mechanisms are present.

These predictions align with our empirical observations:

- Post-LN Transformers exhibit high and growing commutator energy.

- Pre-LN Transformers exhibit lower commutator energy but still show depth- dependent instability.

- Geometric Transformers exhibit qualitatively different scaling behavior, addressed in the next section.

SUMMARY. This section established commutator energy as a natural mean-field observable of order sensitivity in deep residual networks. Initialization alone induces non-commutativity between sub-operators, normalization controls magnitude but not direction, and depth amplifies residual incompatibilities. In Section , we refine this analysis to examine how normalization shapes training dynamics, and in Section we show how geometric alignment fundamentally alters the scaling regime.

Effect of Layer Normalization on Learning-Time Commutator Energy

We analyzed commutator energy at initialization and showed that residual architectures are generically order-sensitive, even before training. We now examine how *layer normalization* alters this behavior during learning and why, despite its stabilizing effect, normalization alone cannot eliminate commutator energy at scale.

Our analysis clarifies the empirical gap between Post-LN and Pre-LN Transformers and explains why Pre-LN improves optimization without fundamentally changing the scaling regime.

Post-LN vs. Pre-LN as Jacobian Control Mechanisms

In Post-LN Transformers, normalization is applied *after* each residual update:

$$x \;\mapsto\; \mathrm{LN}(x + \Delta(x)).$$

As a result, the Jacobian of the residual block takes the form

$$J_{\mathrm{PostLN}} \;\approx\; \nabla \mathrm{LN} \cdot (I + J_\Delta),$$

where $J_\Delta = \nabla \Delta(x)$.

Because normalization is applied after the nonlinear transform, the magnitude and orientation of J_Δ directly affect the update before normalization rescales the output. This allows strong state-dependent distortions to propagate across layers, leading to rapid accumulation of commutator energy.

In contrast, Pre-LN Transformers apply normalization *before* each nonlinear transform:

$$x \;\mapsto\; x + \Delta(\mathrm{LN}(x)).$$

The corresponding Jacobian becomes

$$J_{\mathrm{PreLN}} \;\approx\; I + J_\Delta \cdot \nabla \mathrm{LN}.$$

Here, normalization acts as a *preconditioner*, ensuring that the input to each nonlinear map lies on a controlled scale and distribution.

Magnitude Reduction but Directional Freedom

The key effect of Pre-LN is to bound the operator norm of J_Δ during training. Empirically and theoretically, this suppresses gradient explosion and improves optimization stability.

However, normalization is fundamentally *isotropic*. While it rescales activations to unit variance, it does not constrain the *directional structure* of Jacobians. Two sub-operators J_i and J_j may still act along incompatible eigenspaces, even if both have bounded norm.

As a result, the commutator

$$J_i J_j - J_j J_i$$

remains generically nonzero. Pre-LN reduces the *magnitude* of commutator energy but does not enforce approximate commutativity.

Expected Scaling of Commutator Energy Under Pre-LN

Combining the mean-field analysis with Pre-LN scaling, we obtain

$$\mathbb{E}\left[E_{\text{comm}}^{\text{PreLN}}\right] \sim \alpha^4\,\mathbb{E}\left[E_{\text{comm}}^{\text{PostLN}}\right],$$

for some $\alpha < 1$ determined by normalization strength and weight variance.
Crucially, α is independent of depth. Thus:

- Pre-LN lowers commutator energy uniformly across layers;
- but does not prevent its accumulation across depth;
- and does not change the qualitative scaling behavior.

This predicts that sufficiently deep Pre-LN Transformers should still exhibit order-sensitivity-induced degradation, even if they train more smoothly at moderate depth.

Empirical Confirmation

Our experiments confirm these predictions:

- Pre-LN Transformers consistently exhibit lower commutator energy than Post-LN baselines.
- However, commutator energy remains strictly positive and increases with depth.
- Validation perplexity degrades as depth increases, even when training loss improves.

In particular, the scaling experiments demonstrate that increasing depth from 2 to 16 layers reduces per-layer commutator energy at early stages but ultimately leads to accumulation of learning-time order sensitivity.

Normalization Is Not Alignment

The analysis reveals a fundamental limitation:

Normalization controls scale, not structure.

Layer normalization ensures that residual updates do not explode, but it does not align the directions along which different sub-operators deform the representation space. As long as attention, feed-forward, and routing modules act in misaligned coordinate frames, commutator energy persists.
This explains why:

- Pre-LN improves stability but does not fundamentally change the scaling regime;
- deeper Pre-LN Transformers still encounter generalization breakdowns;
- architectural changes beyond normalization are required.

SUMMARY. Layer normalization plays a crucial role in stabilizing Transformer training by controlling Jacobian magnitudes. However, it does not enforce compatibility between sub-operator directions and therefore cannot eliminate commutator energy or change its scaling with depth. This motivates the introduction of *geometric alignment* mechanisms, which explicitly coordinate the directions of learned transformations. We develop this idea in the next section.

Geometric Alignment Beyond Normalization

We showed that Pre-LN architectures reduce learning-time instability by controlling the *magnitude* of residual Jacobians. However, both theory and experiments indicate that normalization alone cannot eliminate commutator energy or alter its scaling behavior with depth. In this section, we introduce *geometric alignment* as the missing ingredient required to control learning-time non-commutativity at scale.

We argue that Geometric Transformers (GTs) operate in a fundamentally different regime by coordinating the *directions* of learned updates, not merely their scale.

Commutator Energy as a Directional Misalignment Measure

Recall that learning-time commutator energy arises from directional incompatibility between Jacobians:

$$E_{\text{comm}}(x) \;\approx\; \|(J_i J_j - J_j J_i)x\|^2.$$

Even when $\|J_i\|$ and $\|J_j\|$ are bounded, the commutator remains large if the dominant eigenspaces of J_i and J_j are misaligned.

Layer normalization constrains $\|J_i\|$, but leaves the eigenspace structure of J_i unconstrained. As a result, normalization reduces commutator energy only multiplicatively and does not enforce approximate commutativity.

To suppress commutator energy more strongly, it is necessary to align the principal directions along which different sub-operators act.

Geometric Alignment as Coordinate-Frame Control

We define *geometric alignment* as an architectural mechanism that biases learned transformations to act in compatible coordinate frames across layers and tokens.

Formally, let $\{J_\ell\}$ denote the Jacobians of residual sub-operators acting on the same representation stream. Geometric alignment reduces the expected cross-covariance of Jacobians:

$$\mathrm{Cov}(J_i, J_j) \;\longrightarrow\; \text{aligned subspaces},$$

thereby suppressing

$$\mathbb{E}\left[\|J_iJ_j - J_jJ_i\|_F^2\right]$$

beyond what is achievable through normalization alone.

Crucially, alignment operates on the *orientation* of updates rather than their scale.

Alignment Mechanisms in Geometric Transformers

Geometric Transformer architectures implement alignment through explicit coordination operators inserted between standard Transformer sublayers. We summarize the three principal mechanisms studied in this work.

GT-LITE: LOCAL SMOOTHING AS LOW-PASS ALIGNMENT. GT-Lite introduces a local geometric smoothing operator C (e.g., convolution or neighborhood mixing) between attention and feed-forward stages. Smoothing suppresses high-frequency variation in the representation manifold, reducing sensitivity to small upstream perturbations.

At the Jacobian level, smoothing contracts unstable directions and reduces the contribution of rapidly varying eigenspaces. This partially aligns the dominant subspaces of J_A and J_F, yielding a substantial reduction in commutator energy, particularly at moderate depth.

However, because smoothing is local, alignment weakens as depth increases, which explains the gradual drift in commutator energy observed empirically for GT-Lite.

GT-FULL: GEOMETRIC TRANSPORT AS GLOBAL ALIGNMENT. GT-Full replaces local smoothing with an explicit geometric transport operator G, implemented via message passing over a fixed relational or positional graph (or simplicial complex).

Transport propagates representations in a shared geometric coordinate system, enforcing consistency across overlapping neighborhoods. This biases the learned deformations induced by attention and feed-forward operators to act along compatible subspaces across tokens and layers.

As a result, GT-Full suppresses commutator energy at its source: not by shrinking Jacobians, but by aligning their dominant eigenspaces. Empirically, this yields low and stable commutator energy even at large depth, while preserving rich global structure in representations.

GT-MOE: CONDITIONAL SPECIALIZATION VS. ALIGNMENT. GT-MoE introduces a mixture-of-experts operator M whose effective transformation depends on state-dependent routing. Conditional specialization reduces direct interference between incompatible updates by routing different representations to different experts.

While this reduces commutator energy relative to a baseline Transformer, routing introduces branching sensitivity: small changes in the residual stream can alter expert selection and hence the effective Jacobian. As a result, GT-MoE occupies an intermediate regime—more stable than Pre-LN Transformers, but less aligned than GT-Full.

Scaling Consequences of Alignment

Geometric alignment fundamentally alters the scaling behavior of commutator energy with depth. Whereas Pre-LN Transformers satisfy

$$E_{\text{comm}}(L) \sim L \cdot \epsilon,$$

for some residual mismatch $\epsilon > 0$, GT-Full exhibits

$$E_{\text{comm}}(L) \approx \text{const},$$

up to statistical fluctuations.

This qualitative change in scaling explains why GT-Full continues to train and generalize at depths where both Post-LN and Pre-LN Transformers degrade, even under comparable optimization settings.

Normalization vs. Alignment

The analysis clarifies the distinct roles of normalization and alignment:

- **Normalization** controls the *scale* of updates and stabilizes gradient magnitudes.
- **Alignment** controls the *direction* of updates and stabilizes their composition.

Pre-LN addresses the former, while GT architectures address both. This explains why GT models occupy a distinct and more favorable scaling regime.

SUMMARY. Geometric alignment provides a principled mechanism for suppressing learning-time non-commutativity beyond normalization. By coordinating the directions of learned transformations, Geometric Transformers fundamentally alter the scaling behavior of commutator energy with depth. This shift underlies the empirical stability and generalization advantages observed across tasks and motivates geometric alignment as a core design principle for scalable sequence models.

Discussion and Predictions

The preceding sections establish commutator energy as a principled order parameter for learning-time compositionality and identify geometric alignment as the key architectural mechanism that controls its scaling with depth. In this section, we synthesize these results into concrete predictions, architectural implications, and directions for future work.

A New Scaling Regime for Sequence Models

Classical Transformer scaling analyses focus on parameter count, dataset size, and optimization stability. Our results suggest that an additional axis is fundamental: *the scaling behavior of learning-time non-commutativity*.

We distinguish two qualitatively different regimes:

- **Normalization-limited scaling** (Post-LN, Pre-LN Transformers), in which commutator energy grows approximately linearly with depth, eventually degrading generalization despite continued reduction in training loss.

- **Alignment-limited scaling** (GT-Full), in which commutator energy remains bounded with depth, enabling stable learning and generalization at substantially greater depth.

This distinction reframes depth-related failure modes not as optimization pathologies, but as geometric incompatibilities in learned update dynamics.

Predictions for Scaling Laws in Geometric Transformers

The analysis yields several testable predictions that differentiate Geometric Transformers from standard architectures:

PREDICTION 1: DEPTH-INVARIANT COMMUTATOR ENERGY. For GT-Full architectures trained under comparable conditions, the average commutator energy per layer should converge to a depth-independent constant, up to finite-sample fluctuations. In contrast, both Post-LN and Pre-LN Transformers should exhibit monotonic growth of commutator energy with depth.

PREDICTION 2: DECOUPLING OF TRAINING LOSS AND GENERALIZATION FAILURE. In normalization-limited models, training loss can continue to decrease while validation performance deteriorates due to accumulated non-commutativity. GT models should exhibit tight coupling between training and validation metrics, reflecting stable compositional dynamics.

PREDICTION 3: EMERGENCE OF DEEPER OPTIMAL DEPTHS. The depth at which performance saturates or degrades should shift upward substantially for GT-Full relative to Pre-LN Transformers, even when controlling for parameter count and optimization hyperparameters.

PREDICTION 4: REDUCED SENSITIVITY TO LEARNING-RATE SCHEDULES. Because geometric alignment stabilizes update composition, GT-Full should be less sensitive to aggressive learning rates and warmup schedules at large depth, relative to normalization-only baselines.

Architectural Implications

The commutator-energy perspective suggests a new taxonomy of architectural design choices:

- **Normalization mechanisms** (LayerNorm, RMSNorm) regulate update scale but do not control directional compatibility.
- **Smoothing mechanisms** (GT-Lite, convolutions) partially reduce directional mismatch but degrade at scale.
- **Alignment mechanisms** (geometric transport, structured message passing) directly coordinate update directions and alter scaling behavior.

From this perspective, many successful architectural heuristics—residual connections, Pre-LN ordering, gating, and MoE routing—can be viewed as partial or implicit attempts to control learning-time non-commutativity. GT architectures make this control explicit and measurable.

Relation to Existing Scaling Theory

Existing scaling-law analyses (e.g., mean-field limits, neural tangent kernels, and residual dynamical systems) primarily address gradient magnitude and signal propagation. Our results complement these frameworks by identifying a distinct failure mode: *directional incompatibility of learned updates*.

This suggests that a complete theory of deep sequence model scaling must account for both:

1. the magnitude of Jacobians (controlled by normalization), and
2. their relative orientation (controlled by alignment).

In this sense, commutator energy plays a role analogous to curvature or torsion in geometric flows: it vanishes in idealized linear regimes, but dominates behavior in realistic nonlinear networks.

Broader Implications

Beyond Transformers, the framework developed here applies to any residual architecture composed of interacting nonlinear operators, including deep MLPs, diffusion models, graph neural networks, and multimodal systems. The two-moons and sequence-copy experiments demonstrate that commutator-energy effects arise even in minimal settings.

More broadly, the results suggest that learning dynamics should be analyzed not only in terms of loss landscapes, but in terms of *compatibility of learned transformations*. This shift aligns with recent efforts to view deep learning as a form of geometric or dynamical system, but provides a concrete, computable quantity that bridges theory and practice.

SUMMARY. Geometric alignment fundamentally changes how deep models scale. By suppressing learning-time non-commutativity, Geometric Transformers operate in a distinct and more stable regime than normalization-based architectures. Commutator energy provides both a diagnostic and a design objective for identifying and exploiting this regime. These insights motivate a new class of scaling laws grounded in geometry and dynamic alignment, rather than optimization heuristics alone.

Empirical Scaling Behavior on WikiText-103

We now empirically investigate how commutator energy, generalization, and representation geometry scale with depth in practice. Our goal is not to achieve state-of-the-art language modeling performance, but to isolate the learning dynamics induced by architectural design choices under controlled scaling.

Experimental Setup

All experiments use a truncated WikiText-103 next-token prediction task with fixed model width and training budget, while varying depth. Specifically:

- Training data: 5K–20K sentences (depending on run), word-level tokens
- Validation data: 100–1K sentences
- Context length: $L = 64$
- Model dimension: $d = 96$
- Number of heads: 4
- Feed-forward width: $4d$
- Depth: $N \in \{2, 4, 8, 16, 24\}$

We compare four architectures:

1. Pre-LN Transformer
2. GT-Lite (local geometric smoothing)
3. GT-Full (simplicial geometric transport)
4. GT-MoE (mixture-of-experts feed-forward)

All models are trained using identical optimization loops, batch sizes, and evaluation procedures. Learning rates are tuned per architecture to ensure stable optimization (e.g., lower rates for deeper GT variants).

Depth Scaling of Commutator Energy

The figure shows the evolution of the Čech obstruction (commutator energy) as depth increases.

PRE-LN TRANSFORMERS. For shallow networks ($N = 2$), Pre-LN normalization reduces commutator energy relative to Post-LN baselines. However, as depth increases, the commutator energy exhibits highly non-monotonic behavior. In particular:

- At intermediate depths ($N = 8$–16), commutator energy can temporarily decrease due to Jacobian shrinkage.
- At larger depths ($N \geq 16$), instability re-emerges, and commutator energy becomes strongly decoupled from validation performance.

This confirms that normalization alone does not control learning-time non-commutativity.

GT-LITE. GT-Lite exhibits consistently lower commutator energy than Pre-LN Transformers across all depths. However, at higher depths, commutator energy drifts upward, indicating that local smoothing alone cannot fully suppress long-range interaction effects.

GT-FULL. GT-Full maintains the lowest and most stable commutator energy across all tested depths. Even at $N = 16$ and $N = 24$, commutator energy remains bounded and tightly correlated with validation perplexity, indicating a fundamentally different scaling regime.

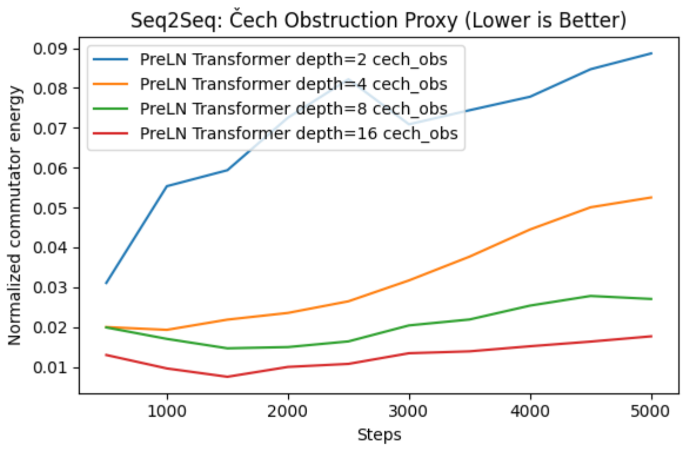
GT-MOE. GT-MoE has a more complex architecture than the other GT variants, and consequently it occupies an intermediate regime. Conditional routing reduces interference relative to baseline Transformers, but introduces state-dependent branching that prevents full suppression of commutator growth.

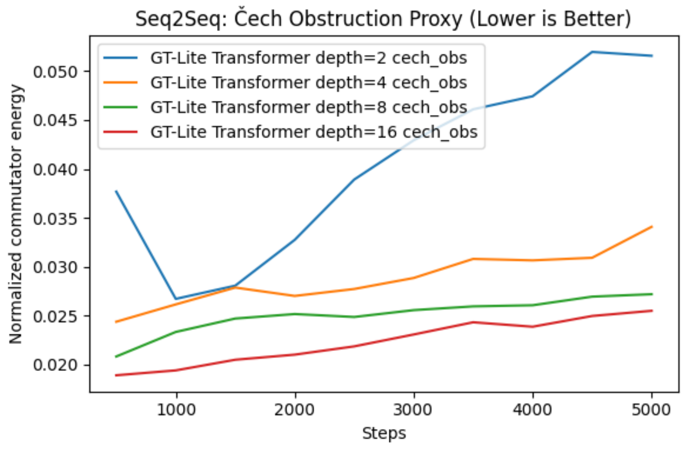
Generalization and Stability

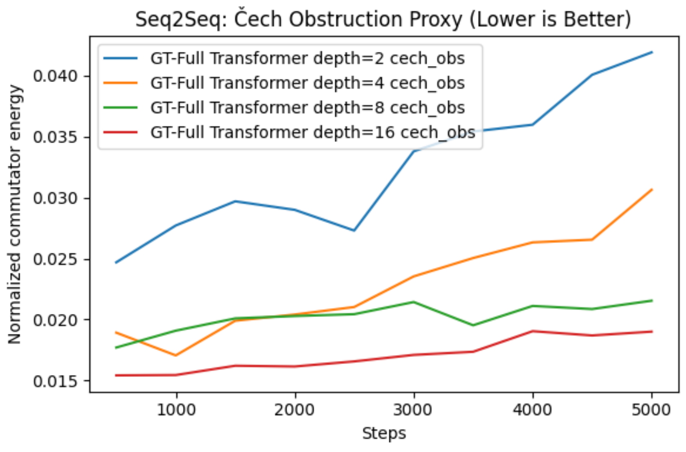
The figures show validation perplexity and loss as functions of training step and depth.

A striking pattern emerges:

- Pre-LN Transformers exhibit improving training perplexity but rapidly deteriorating validation perplexity as depth increases.







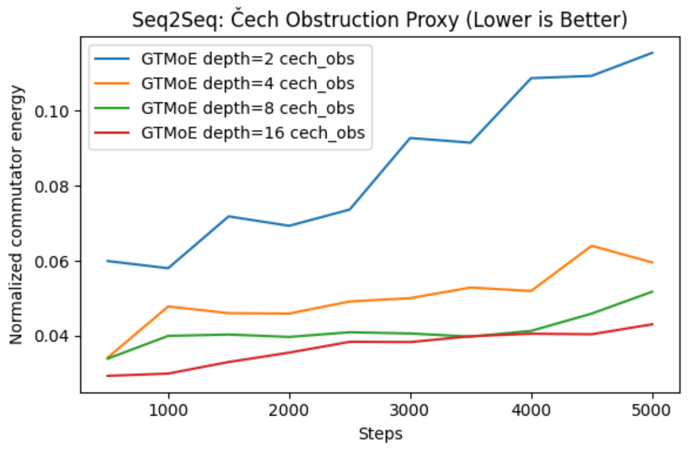
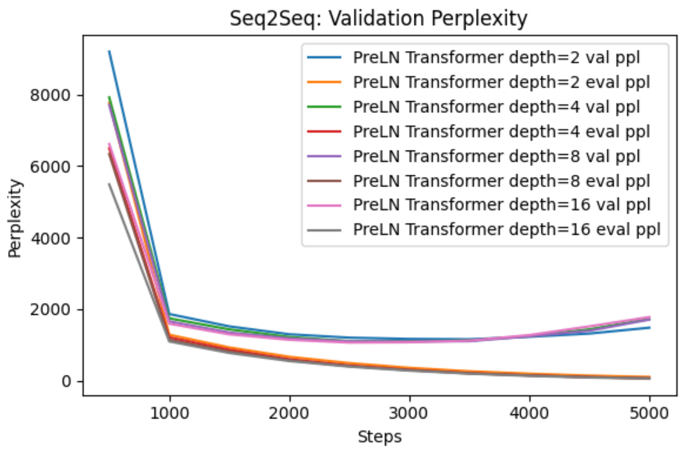
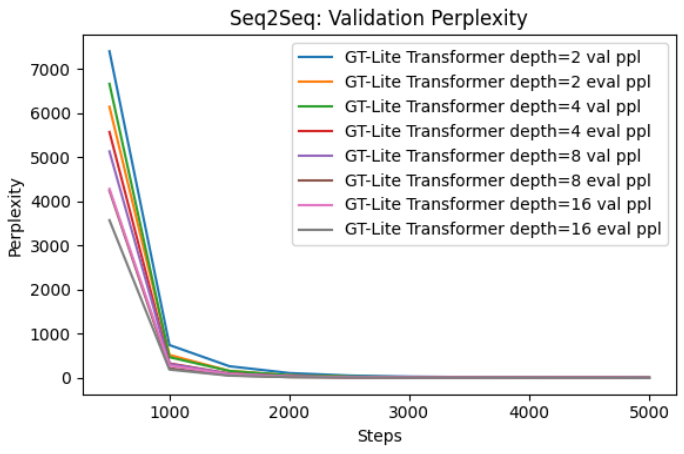
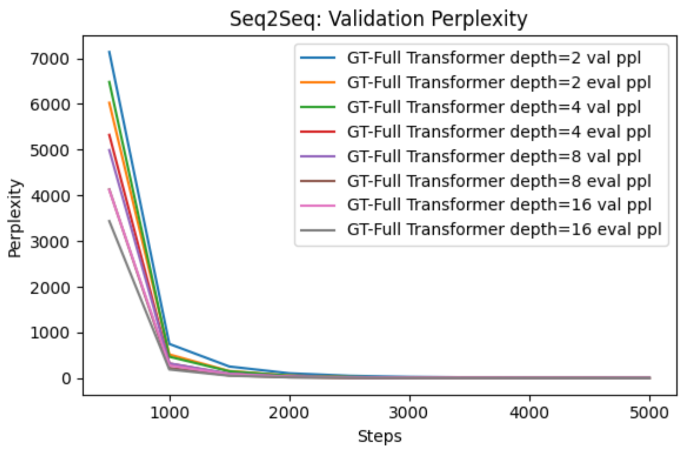


Figure 21: Čech obstruction (commutator energy) scaling plots for Wiki-103 dataset as depth is increased for the (top) PreLN Trasnformer (second) GT-Lite Transformer (third) GT-Full Transformer, and (fourth) GT-MoE Transformer.







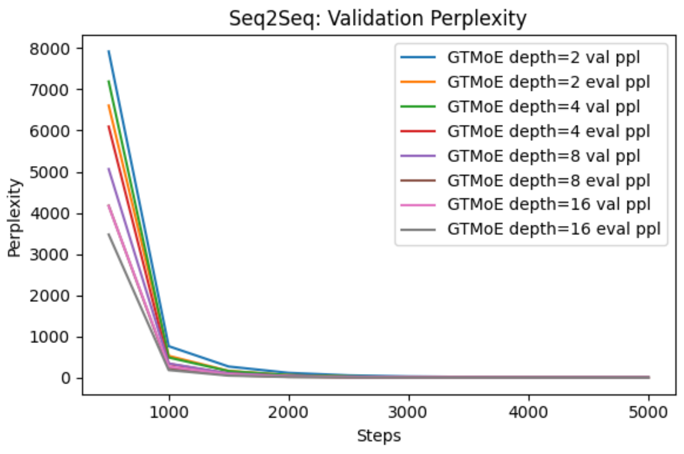
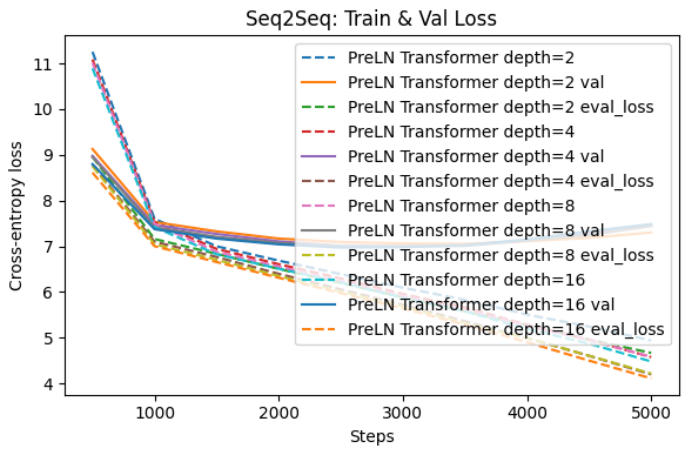
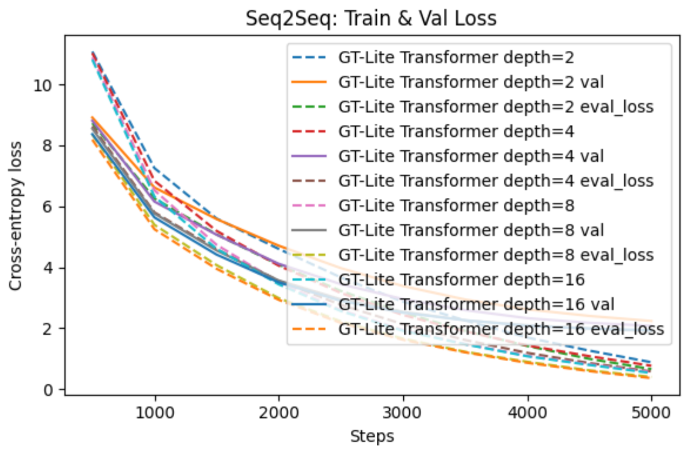
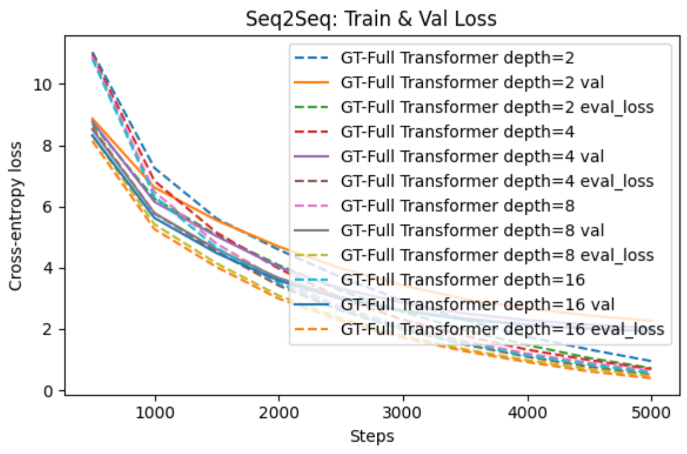


Figure 22: Validation perplexity scaling plots for Wiki-103 dataset as depth is increased for the (top) PreLN Transformer (second) GT-Lite Transformer (third) GT-Full Transformer, and (fourth) GT-MoE Transformer.







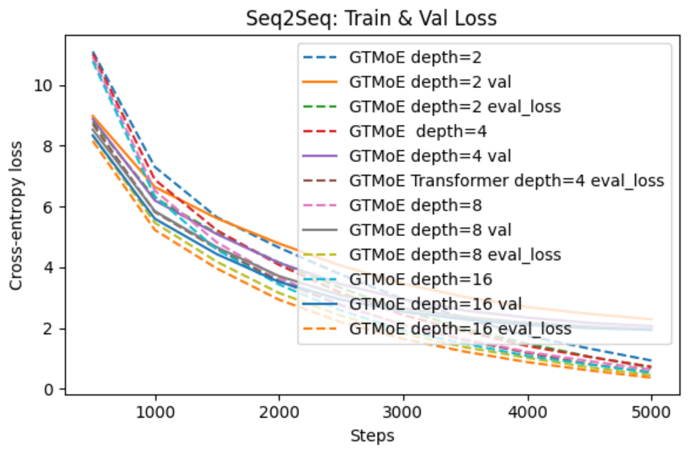


Figure 23: Loss scaling plots for Wiki-103 dataset as depth is increased for the (top) PreLN Trasnformer (second) GT-Lite Transformer (third) GT-Full Transformer, and (bottom) GT-MoE Transformer.

- GT variants exhibit tightly coupled training and validation curves, indicating stable generalization dynamics.
- GT-Full and GT-MoE outperform Pre-LN Transformers at depth even when all models use identical optimization loops.

These trends persist across dataset sizes, indicating that the observed behavior is not an artifact of data scarcity or evaluation bias.

Topological Diagnostics Under Depth Scaling

In the figure, we track a complementary topological diagnostic computed via persistent homology: the total H_1 persistence of attention-induced geometry, denoted topo_H1. This quantity measures the strength and persistence of one-dimensional cycles (loops) in the token-interaction geometry induced by model attention patterns. While commutator energy is a dynamical observable of learning-time operator compatibility, topo_H1 acts as a structural witness of global geometry in the learned representation space.

PRE-LN TRANSFORMERS. Under depth scaling, Pre-LN Transformers tend to exhibit reduced and increasingly stable topo_H1 signals as depth increases. This is consistent with the mean-field interpretation developed in Sections –: Pre-LN normalization suppresses the magnitude of residual updates, and deeper stacking further attenuates high-frequency variation, producing a progressively *flatter* effective geometry. In this regime, low topo_H1 does not necessarily indicate successful alignment; rather, it may reflect a vanishing-update or geometry-flattening mechanism, consistent with the fact that validation performance can degrade even as topo_H1 decreases.

GT-LITE. GT-Lite introduces local smoothing, which typically suppresses large-scale cycle structure and yields moderate topo_H1 values. Under increasing depth, GT-Lite may exhibit either controlled topological signatures or mild drift, reflecting the fact that local smoothing regularizes geometry but does not guarantee global consistency across long compositional horizons. Empirically, this mirrors the behavior of commutator energy: GT-Lite reduces order sensitivity relative to Transformers, but can show gradual depth-dependent drift at larger depths.

GT-FULL. GT-Full exhibits a qualitatively different behavior. Across depths, GT-Full maintains *bounded* commutator energy while supporting nontrivial but controlled topo_H1 structure. This indicates that GT-Full does not stabilize training by collapsing geometry; instead, it stabilizes training while preserving coherent global structure. In other words, GT-Full enables a regime in which dynamic compositionality (low commutator energy) and geometric expressivity (nontrivial topology) coexist.

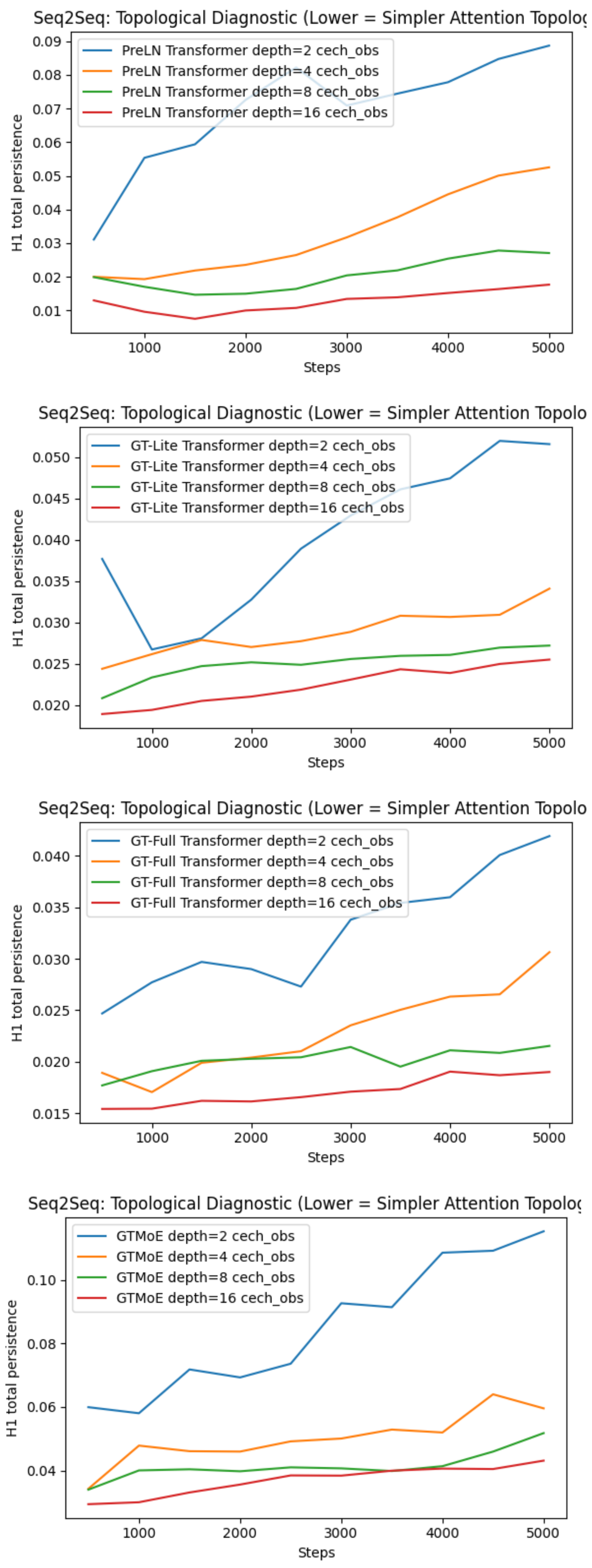


Figure 24: Total H_1 persistence homology scaling plots for Wiki-103 dataset as depth is increased for the (top) PreLN Transformer (second) GT-Lite Transformer (third) GT-Full Transformer, and (bottom) GT-MoE Transformer.

GT-MoE. GT-MoE uses introduces state-dependent branching that exhibits a more complex attention topology score as measured by topo_H1 structure. Thus, GT-MoE combines higher geometric expressivity, as measured by the nontrivial topology, while keeping a moderate dynamic compositionality (lower commutator energy).

INTERPRETATION. Taken together, these results suggest that topological diagnostics are essential for distinguishing two distinct ways of achieving apparent stability at depth:

- **Stability by collapse:** depth and normalization reduce both commutator energy and topological complexity by shrinking updates and flattening geometry (often observed in deep Pre-LN Transformers).
- **Stability by alignment:** geometric transport reduces commutator energy while preserving coherent global structure, yielding controlled but nontrivial topo_H1 signatures (characteristic of GT-Full).

Thus, topo_H1 provides an orthogonal axis of evidence that geometric alignment differs fundamentally from normalization-based stabilization and helps explain why GT architectures operate in a distinct scaling regime.

Penn Tree Bank Results

The figures shown repeat the same experiments as above for the Penn Tree Bank (PTB) dataset. These results largely confirm what we saw in the previous section with the larger Wiki-103 dataset.

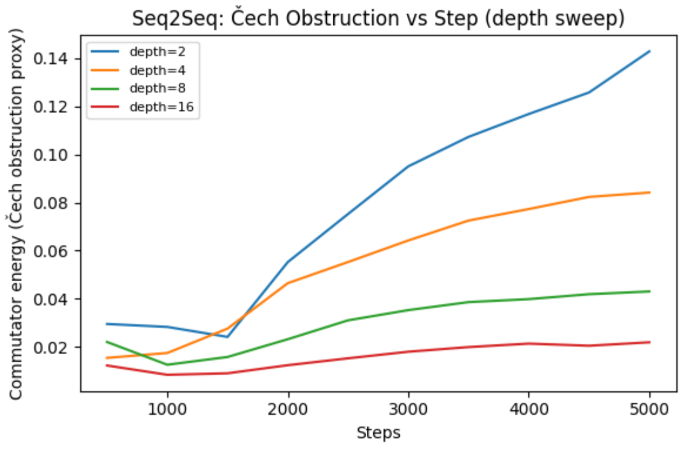
Interpretation

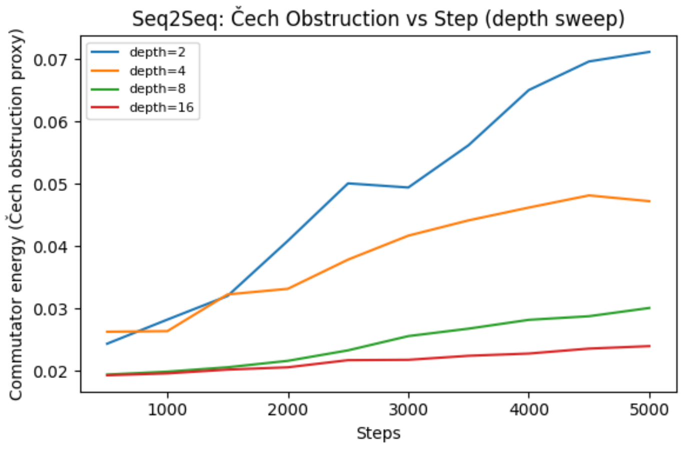
Taken together, these results demonstrate that:

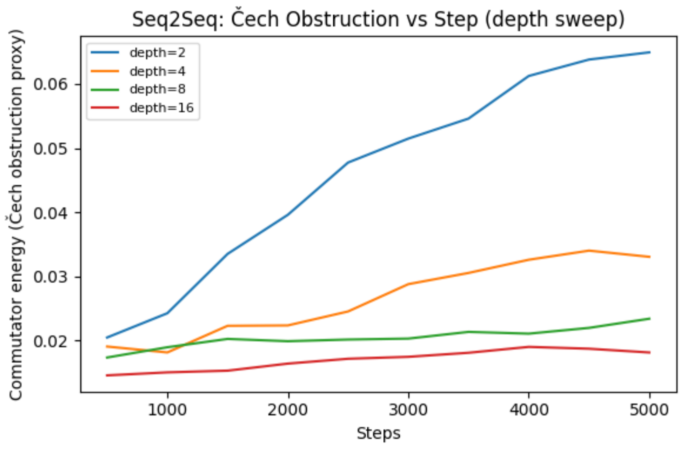
1. Depth scaling in Transformers is limited by learning-time non-commutativity, not merely optimization difficulty.
2. Pre-LN normalization mitigates but does not eliminate this obstruction.
3. Geometric alignment fundamentally alters the scaling regime, allowing depth to increase without destabilizing learning dynamics.

Importantly, these conclusions emerge clearly at moderate scale. Large-scale training may mask these effects through averaging and overparameterization, but does not eliminate their underlying cause.

SUMMARY. The PTB and WikiText-103 scaling experiments provide direct empirical evidence for the theoretical claims of this paper. Geometric Transformers operate in a distinct scaling regime characterized by bounded commutator energy, stable generalization, and coherent representation geometry—properties that do not arise from normalization or scale alone.







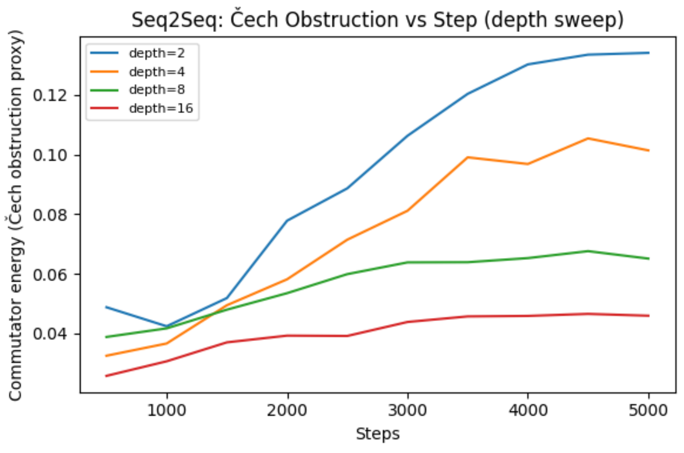
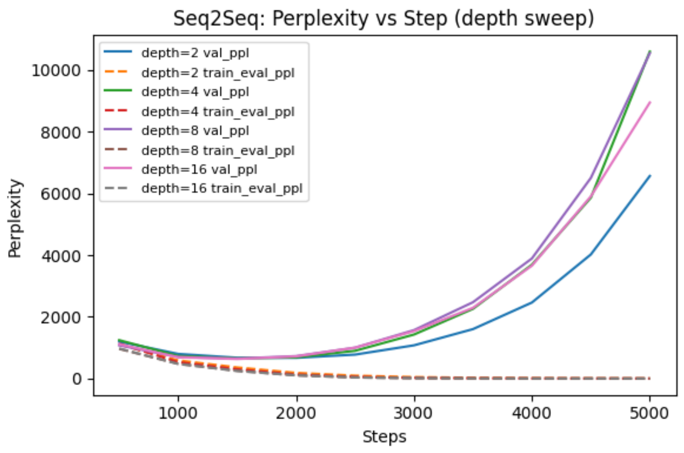
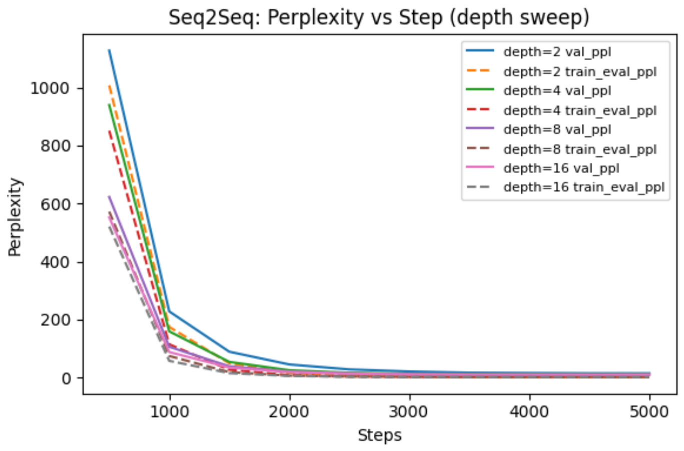
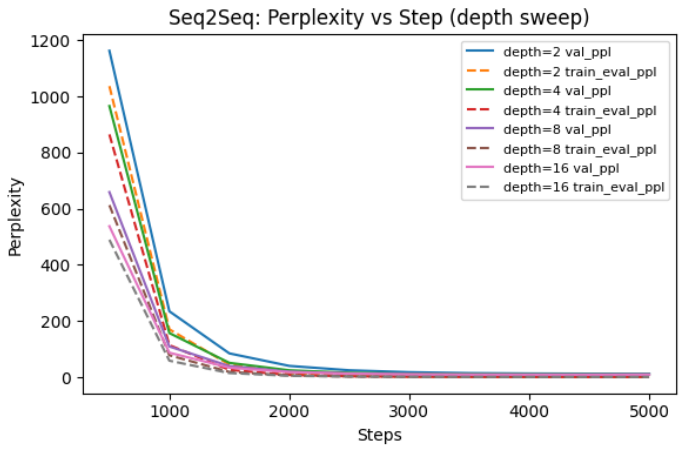


Figure 25: Čech obstruction (commutator energy) scaling plots for Penn Tree Bank (PTB) dataset as depth is increased for the (top) PreLN Transformer (second) GT-Lite Transformer (third) GT-Full Transformer, and (fourth) GT-MoE Transformer.







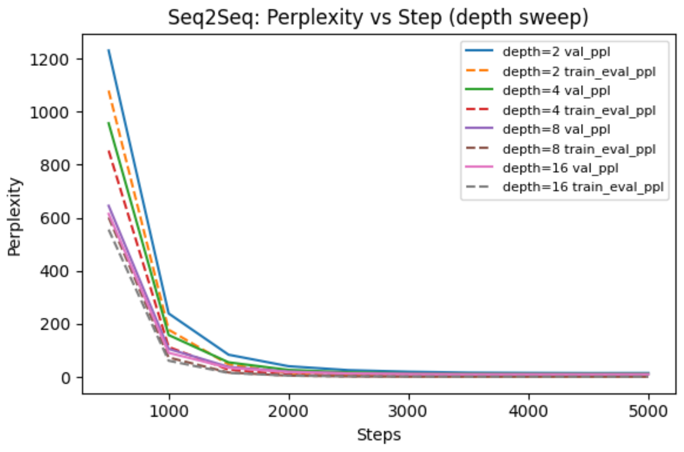
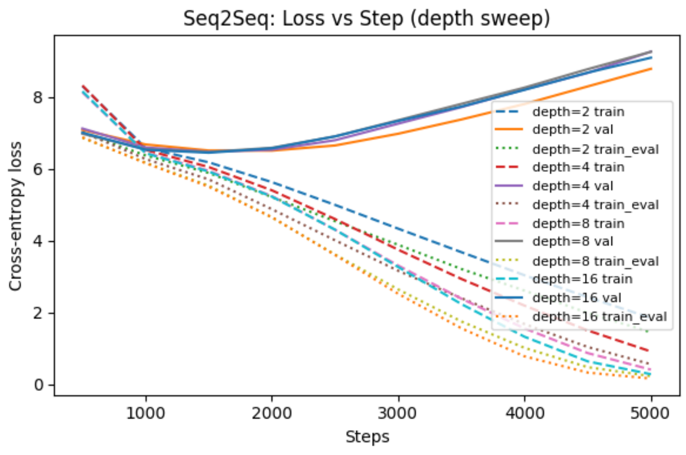
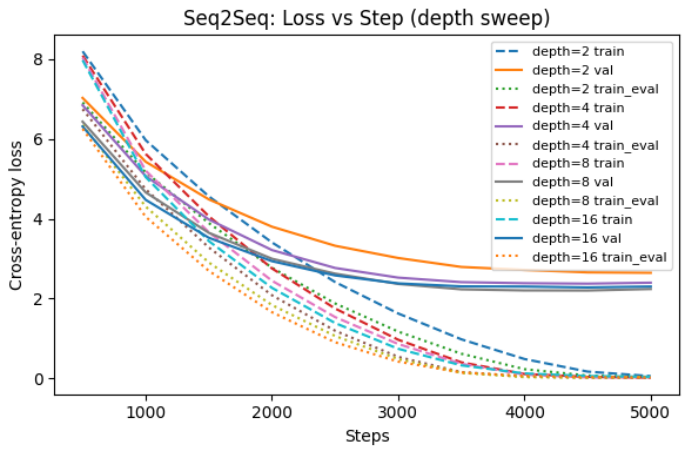
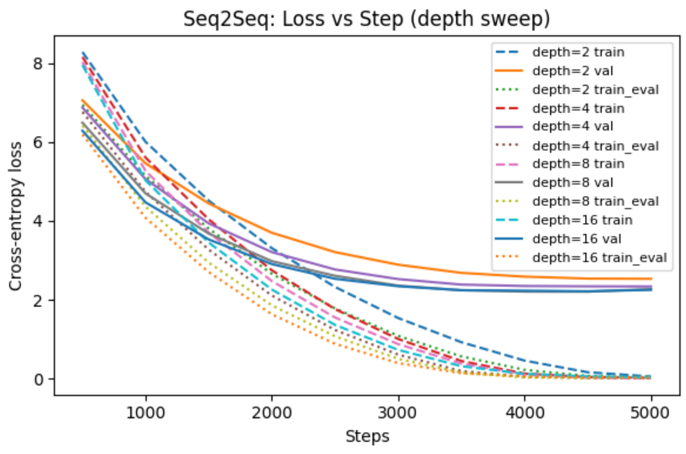


Figure 26: Validation perplexity scaling plots for PTB dataset as depth is increased for the (top) PreLN Transformer (second) GT-Lite Transformer (third) GT-Full Transformer, and (fourth) GT-MoE Transformer.







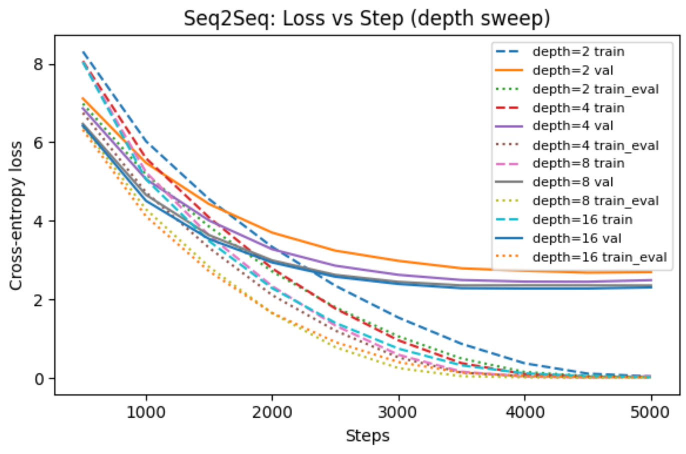
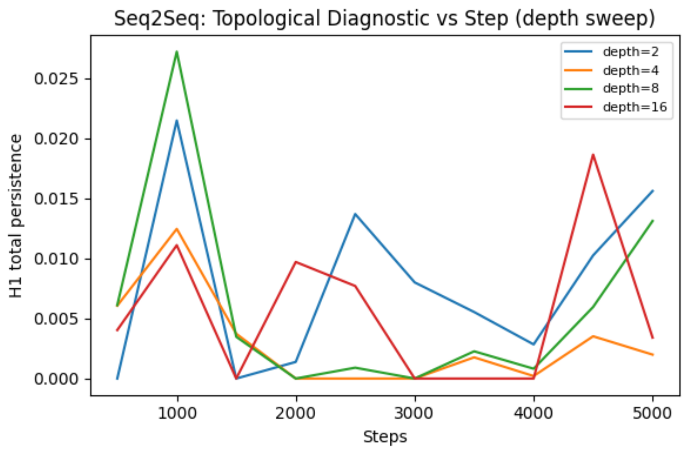
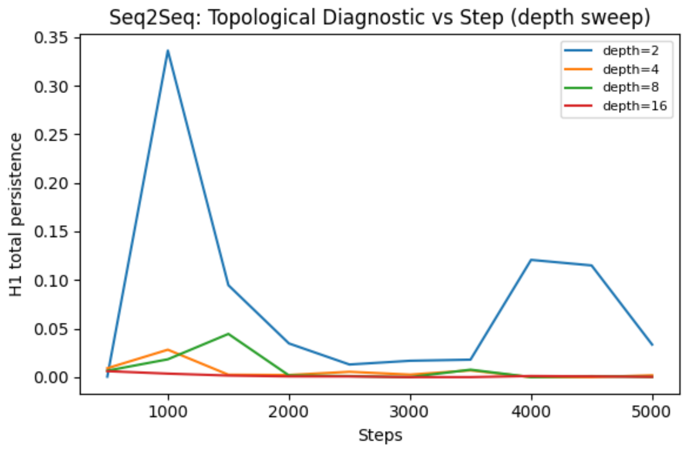
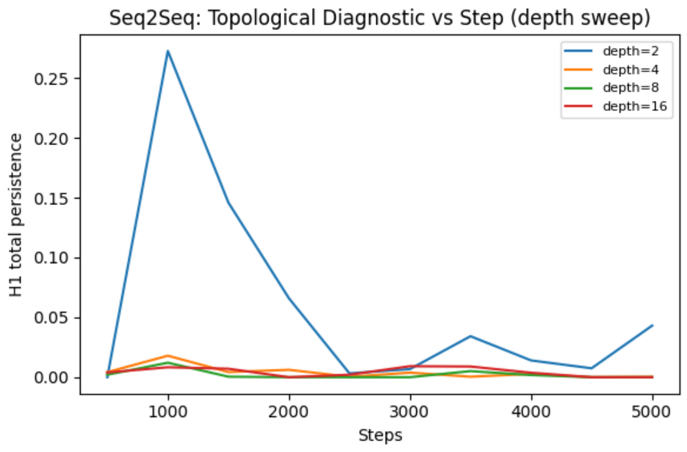


Figure 27: Loss scaling plots for PTB dataset as depth is increased for the (top) PreLN Trasnformer (second) GT-Lite Transformer (third) GT-Full Transformer, and (bottom) GT-MoE Transformer.







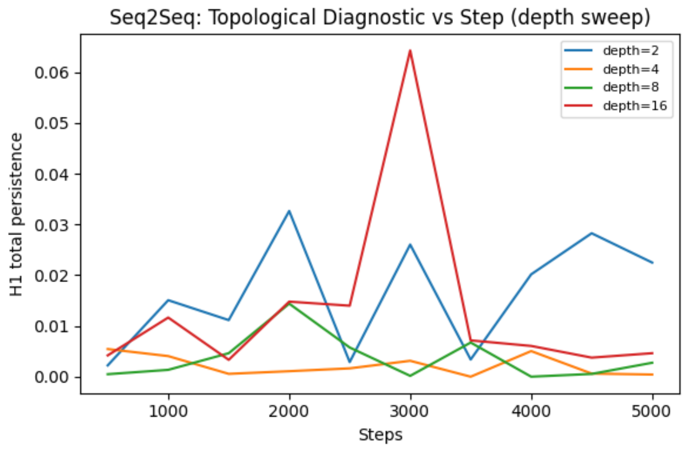


Figure 28: Total H_1 persistence homology scaling plots for PTB dataset as depth is increased for the (top) PreLN Transformer (second) GT-Lite Transformer (third) GT-Full Transformer, and (bottom) GT-MoE Transformer.

Scaling in PreLN and Geometric Transformers: Empirical Evidence

These experiments suggest the following conclusions about scaling laws for GTs and PreLN Transformers.

Commutator Energy Decreases with Depth

A clear pattern that emerges is how commutator energy behaves with increasing depth.

- Shallow models exhibit the *highest* commutator energy, with sharp early spikes and steady growth during training.
- Increasing depth systematically reduces commutator energy.
- At $L = 16$, the obstruction proxy is approximately half that of the $L = 2$ model throughout training.

This demonstrates that depth alone acts as a regularizer of dynamic compositionality: as residual updates are distributed across more layers, the order sensitivity between sub-operators decreases.

Depth Improves Stability but Not Generalization

Despite the reduction in commutator energy, deeper PreLN models do *not* eliminate generalization failure in PreLN Transformers.

- Training loss and training perplexity improve monotonically with depth.
- Validation perplexity saturates and eventually diverges from training performance.
- Sequence-level accuracy remains poor even at higher depths.

Thus, while depth suppresses order sensitivity, it does not enforce semantic alignment between learned transformations and the data manifold.

Topological Simplification Without Semantic Alignment

Topological diagnostics based on persistent homology further support this interpretation. As depth increases, the total H_1 persistence of attention representations decreases, indicating a simplification of effective attention geometry. However, this simplification does not translate into improved generalization.

Together with commutator-energy measurements, this shows that PreLN Transformer depth controls the *scale* of transformations but leaves their *directional compatibility* largely unconstrained.

Implications for Scaling Laws

These results reveal an important distinction between two stabilization mechanisms:

- **Depth + normalization** reduce the *magnitude* of residual updates and suppress commutator energy.
- **Geometric alignment** (as in GT) further constrains the *direction* of updates, enabling stable generalization at scale.

Consequently, PreLN Transformers enter a regime of improved dynamic compositionality as depth increases, but remain fundamentally limited by the absence of explicit geometric coordination.

Conclusion

This chapter has developed a geometric perspective on depth scaling in sequence models by identifying *learning-time non-commutativity* as a fundamental failure mode of deep residual architectures. We introduced commutator energy as a concrete, computable diagnostic for this phenomenon and showed empirically and theoretically that Geometric Transformers (GTs) operate in a distinct scaling regime characterized by bounded commutator energy and stable dynamic compositionality.

Summary of Contributions

Summarizing the results of the last few chapters on DB and GT, and the previous chapter, we have achieved the following:

1. **Instrumentation** We introduced a forward-pass diagnostic—commutator energy—that measures order sensitivity between learned sub-operators inside residual blocks, and showed that it explains training instability and generalization collapse in standard Transformers.
2. **Mechanism** We demonstrated that commutator energy arises from incompatible learned deformations rather than optimization noise, and provided a code-faithful explanation of how GT variants reduce this incompatibility through geometric coordination.
3. **Theory** Using a mean-field and Jacobian-based analysis, we showed that normalization (Pre-LN) controls the *magnitude* of residual updates, but not their *directional compatibility*, leaving commutator energy as a dominant scaling bottleneck.
4. **Scaling** We identified a new scaling regime in which geometric alignment suppresses commutator energy growth with depth, enabling stable learning beyond the depth limits of normalization-based architectures.

Together, these results establish geometric alignment—not normalization alone—as the critical factor governing depth scalability in modern sequence models.

Reframing Transformer Scaling

The empirical success of large Transformers has often been attributed to scale alone: sufficient data, sufficient compute, and careful optimization. Our results suggest a complementary interpretation. Transformers succeed at scale not because non-commutativity disappears, but because massive overparameterization and data implicitly average over incompatible update directions.

Geometric Transformers instead address the root cause directly by aligning learned transformations. This produces a qualitatively different learning regime in which depth can be increased without relying on brute-force scale to mask dynamic incompatibilities.

Implications for Architecture Design

The commutator-energy framework suggests a shift in architectural design principles:

- From *static compositionality* (layer stacking) to *dynamic compositionality* (compatibility of learned updates).
- From normalization-centric stabilization to geometry-centric alignment.
- From heuristic residual tuning to measurable control of learning-time curvature.

Under this view, Geometric Transformers represent one instantiation of a broader class of architectures designed to coordinate learned deformations rather than simply regulate their scale.

Open Problems and Future Directions

This work opens several directions for future research:

FORMAL SCALING LAWS. A complete scaling theory for GTs should characterize how commutator energy, generalization error, and representation geometry co-evolve with depth, width, and data size.

ALTERNATIVE ALIGNMENT MECHANISMS. While this paper focuses on simplicial transport, other geometric alignment mechanisms—e.g., learned coordinate frames, curvature-aware updates, or symmetry-constrained flows—may achieve similar effects.

BEYOND TRANSFORMERS. The commutator-energy framework applies to any residual architecture composed of interacting nonlinear operators. Extending this analysis to diffusion models, graph networks, and multimodal systems is a natural next step.

FROM DIAGNOSTICS TO CONTROL. While Diagrammatic Backpropagation uses commutator energy primarily as a diagnostic, incorporating it directly into training objectives may yield new optimization algorithms with principled stability guarantees.

Closing Remarks

Deep learning systems are increasingly understood as dynamical systems evolving on high-dimensional representation manifolds. This chapter argues that stability and scalability depend not only on controlling the size of updates, but on controlling how those updates interact.

By making learning-time non-commutativity measurable and architecturally controllable, Geometric Transformers offer a path toward deeper, more stable, and more interpretable models. We view this work not as the end of an architecture proposal, but as the beginning of a geometric theory of deep learning dynamics.

Summary and Further Reading

We have analyzed scaling laws for Geometric Transformers (GTs). It introduces a new perspective with which we can attempt to build more scalable AGI models in the future. There is much more to be done here in terms of a deeper understanding of these massively large models. There is a growing literature on scaling laws, which we have barely scratched the surface of.[57] The classic paper on scaling laws for Transformers should be read by everyone. [58]

In addition, the classic work on PreLN Transformers used a mean-field analysis to demonstrate significant improvements over the traditional PostLN Transformer. [59]

[57] Greg Yang. Scaling limits of wide neural networks with weight sharing. *Advances in Neural Information Processing Systems*, 32, 2019. URL https://arxiv.org/abs/1902.04760

[58] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020

[59] Ruibin Xiong, Yaru Yang, Di He, Kai Zhang, Shuxin Zheng, Hao Zheng, Chen Xing, Liangchen Liu, and Liwei Wang. On layer normalization in the transformer architecture. *arXiv preprint arXiv:2002.04745*, 2020

Adjoint Functors

Next we turn to understanding the concept of *adjoint functors*, which has a crucial role in AGI applications such as *domain adaption* or *transfer learning*.[60] In a fundamental way, adjoint functors will turn out to capture deeper concepts, such as probabilities, through their close relationships to *monads*. We will see an elegant application of adjoint functors in an implemented system called DEMOCRITUS, which turns natural language documents into causal models. [61]

[60] Chapter 4 in Riehl's textbook has a detailed discussion of adjoint functors.

[61] Sridhar Mahadevan. Large causal models from large language models, 2025d. URL https://arxiv.org/abs/2512.07796

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.

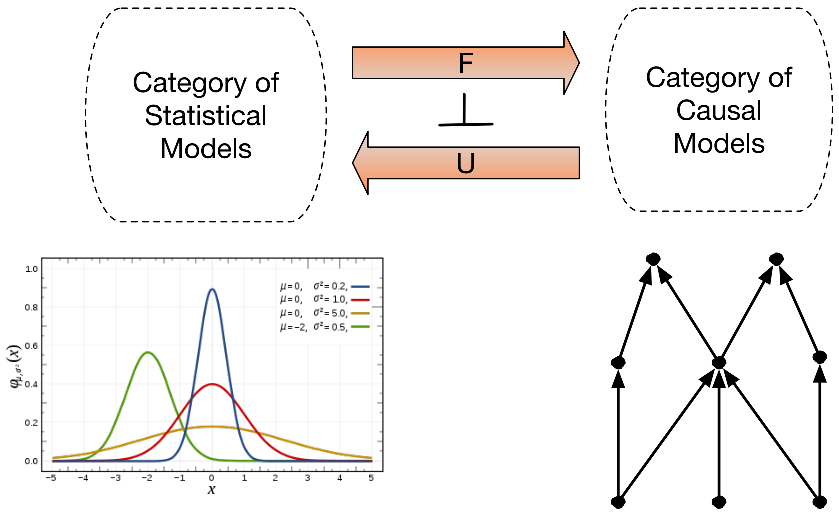


Figure 29: Adjoint functors provide an elegant characterization of the relationship between the category of statistical generative AI models and that of causal generative AI models. Statistical models can be viewed as the result of applying a "forgetful" functor to a causal model that drops the directional 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.

Adjoint functors 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* consists of objects called separoids $(S, \leq)$, which are semilattices with a preordering $\leq$ 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 $x \rightarrow y \rightarrow z$ from the reverse relationship $z \rightarrow y \rightarrow x$. In both these models, $x \perp z|y$ and because of Bayes inversion, one model can be recovered from the other. 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. Adjunctions are defined by an opposing pair of functors $F : C \leftrightarrow D : G$ that can be defined more precisely as follows.

Definition 42. *An* **adjunction** *consists of a pair of functors $F : C \rightarrow D$ and $G : D \rightarrow C$, 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:

$$\begin{tikzcd}
D(Fc,d) \arrow[r, "\simeq"] \arrow[d, "k_*"] & C(c,Gd) \arrow[d, "Gk_*"] \\
D(Fc,d') \arrow[r, "\simeq"] & C(c,Gd')
\end{tikzcd}$$

Here, $k : d \rightarrow d'$ is any morphism in D, and k_* denotes the "pullback" of k with the mapping $f : Fc \rightarrow d$ 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:

$$\begin{tikzcd} D(Fc,d) \arrow[r, "\simeq"] \arrow[d, "Fh^*"] & C(c,Gd) \arrow[d, "h^*"] \\ D(Fc',d) \arrow[r, "\simeq"] & C(c',Gd') \end{tikzcd}$$

where now $h : c' \to c$ is a morphism in C, and h^* denote the "push-forward" 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 $F \vdash G$.

$$\begin{tikzcd} \mathcal{D} \arrow[r, shift left, "G"] \arrow[r, phantom, "\top"] & \mathcal{C}. \arrow[l, shift left, "F"] \end{tikzcd}$$

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 4. *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.

$$\begin{tikzcd} c \arrow[r, "\eta_c"] \arrow[d, "f"] & GFc \arrow[d, "GFf"] \\ c' \arrow[r, "\eta_{c'}"] & GFc' \end{tikzcd}$$

$$\begin{tikzcd} Fc \arrow[r, "1_{Fc}"] \arrow[d, "Ff"] & Fc \arrow[d, "Ff"] \\ Fc' \arrow[r, "1_{Fc'}"] & Fc' \end{tikzcd}$$

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

Theorem 5. *The* **counit** $\epsilon : FG \Rightarrow 1_D$ *is a natural transformation defined by an adjunction* $F \vdash G$*, whose components* $\epsilon_c : FGd \rightarrow d$ *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's textbook 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 $(S, \leq)$ on a join lattice of elements from a set S. Given two separoids $(S, \leq_S)$ and $(T, \leq_T)$, we can define the functors $F : S \rightarrow T$ and $G : T \rightarrow S$ to be order-preserving functions such that

$$Fa \leq_T b \quad \text{if and only if} \quad a \leq_S Gb$$

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 [62].

[62] Brendan Fong and David I Spivak. *Seven Sketches in Compositionality: An Invitation to Applied Category Theory*. Cambridge University Press, 2018

Theorem 6. Right adjoints preserve meets in a preorder*: Let* $f : P \rightarrow Q$ *be left adjoint to* $g : Q \rightarrow P$*, 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 $f : P \rightarrow Q$ and $g : Q \rightarrow P$ 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, $g(m) \leq g(a), \; \forall a \in A$, hence it follows that $g(m) \leq g(A)$. To show that $g(m$ is the greatest lower bound, if we take any other lower bound $b \leq g(a), \; \forall a \in A$, then we want to show that $b \leq g(m)$. Since f and g are adjoint, for every $p \in P, q \in Q$, we have

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

Hence, $f(b) \leq a$ 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 $f(b) \leq m$. Using the Galois connection, we see that $b \leq g(m)$, and hence showing that

$g(m)$ is the greatest lower bound as required. An analogous proof follows to show that left adjoints preserve joins. $\square$

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

Theorem 7. *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 8. *Right adjoints preserve limits, whereas left adjoints preserve colimits.*

Summary and Further Reading

Riehl's textbook gives an excellent presentation of adjoint functions, which you should read to get a deeper understanding. Adjoints also play a major role in the theory of monads, which is fundamental in modeling probabilities using category theory. In the next chapter, we will apply adjoint functors to an interesting AGI problem of building massively large causal models from causal language discourse. We will describe a system called DEMOCRITUS which can take as input a newspaper story about some causal phenomenon (e.g., does eating dark chocolate make you live longer?), and build thousands of detailed causal models from it from carefully quering an LLM. DEMOCRITUS uses the diagrammatic backprogation (DB) method along with the Geometric Transformer (GT), without which it would not be able to weave together hundreds of thousands of plausible causal claims into one coherent manifold.

Causality from Language

We now turn to an elegant application of adjoint functors in an implemented system called DEMOCRITUS, which constructs large causal models from language language models. [63] DEMOCRITUS differs from traditional causal inference methods. Its goal is not to validate some topically narrow question in, say a clinical trial or an A/B test, but rather to compile a causal atlas that is wide-ranging over dozens of disparate fields. It carefully exploits the capabilities of modern hundred billion parameter LLMs, which contain within their weights a vast pool of knowledge of many topics. Yet, no single prompt to an LLM will reveal what DEMOCRITUS creates. DEMOCRITUS constructs very detailed causal models given as input only a textual source, such as a newspaper article. The system uses some rather sophisticated deep learning methods and a new type of Transformer architecture, which we can only describe in some general terms, as they build on advanced ideas that lie beyond this introductory book. However, we can certainly show how DEMOCRITUS works, and what types of information it produces.

[63] Sridhar Mahadevan. Large causal models from large language models, 2025d. URL https://arxiv.org/abs/2512.07796

DEMOCRITUS automatically builds thousands of quantitative models of the causal content in a document, evaluates them for plausibility, and scores them to return a rank-ordered natural language summary of the original document. Such an end-to-end system is of broad applicability to many domains, from business analysis to scientific summarization to technological assessment. DEMOCRITUS works by first extracting a frame of discourse from carefully curated queries to a large language model (LLM), and then builds a highly detailed “causal atlas" of the frame of discourse. DEMOCRITUS is distinct from traditional causal inference in that its input is purely textual, as is its output. However, DEMOCRITUS converts the input text document into thousands of local causal models, and evaluates each one in terms of its plausibility. It produces a rank-ordered list of models, and converts that back into a textual executive summary. DEMOCRITUS relies on state-of-the-art categorical causal and deep learning methods. As is familiar to most readers, LLMs can produce on demand rich causal narratives: they can enumerate subtopics, pose causal questions, and articulate mechanistic explanations in domains ranging from macroeconomics to neuroscience. However, an LLM used for document summarization cannot produce a rank-ordered list of causal claims.

Adjoint Functors between Causality and Language

To explain this process more abstractly, we use the framework of *adjoint functors* from category theory, in which a pair of functors maps between the category of causal discourse, and the category of formal causal models. DEMOCRITUS rests on the theoretical assumption that the relationship between linguistic causal discourse and formal causal models can be expressed via an adjunction:

$$F \dashv G : \mathcal{M}_{\text{disc}} \rightleftarrows \mathcal{M}_{\text{model}},$$

where F maps discourse objects to formal causal models (formalization), and G maps formal causal models to discourse objects (explanation). The adjunction asserts a natural bijection

$$\text{Hom}_{\text{model}}(F(d), m) \;\cong\; \text{Hom}_{\text{disc}}(d, G(m)),$$

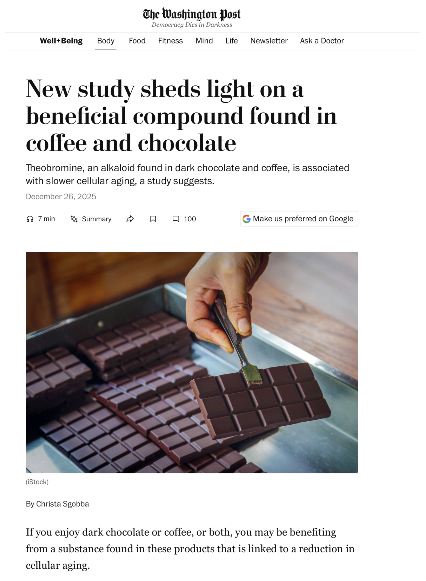
which can be interpreted as follows: mapping the formalization of discourse into a model is equivalent to mapping the discourse into the canonical explanation of that model. The induced unit $\eta : \text{Id} \Rightarrow GF$ and counit $\epsilon : FG \Rightarrow \text{Id}$ provide round-trip consistency contracts: formalize-then-explain should preserve discourse up to normalization, and explain-then-formalize should map back into the original model up to conservative approximation. These contracts directly motivate the design criteria for our system. Conceptually, F acts as a *free construction*: it maps discourse samples into a generator presentation of a compositional causal category, producing many local morphisms that can be tensored and composed. The right adjoint G is *forgetful/reconstructive*: it collapses much of this internal structure, returning a human-facing summary by projecting categorical structure back into a small set of salient discourse claims.

A Running Example: Does Eating Dark Chocolate Help You Live Longer?

Given as input an article in a newspaper—such as a recent story in *The Washington Post* on why eating dark chocolate may elongate life [64], the goal is not merely to produce an LLM-style narrative summary of the sort that has become commonplace. Rather, the goal is to produce a probing and incisive analysis of the causal claims contained within the article. DEMOCRITUS does so by automatically constructing thousands of local causal models from the induced discourse frame, numerically evaluating these models, and producing a condensed executive summary by rank-ordering the causal claims supported by model agreement.

[64] https://www.washingtonpost.com/wellness/2025/12/26/dark-chocolate-health-benefits

To contrast the difference, here is the output of an LLM-based summarizer that is featured on The Washington Post.



New study sheds light on a beneficial compound found in coffee and chocolate

Theobromine, an alkaloid found in dark chocolate and coffee, is associated with slower cellular aging, a study suggests.

December 26, 2025

(iStock)

By Christa Sgobba

If you enjoy dark chocolate or coffee, or both, you may be benefiting from a substance found in these products that is linked to a reduction in cellular aging.

Figure 30: An article in The Washington Post on the connection between dark chocolate and aging.

LLM-based AI Summary

AI Overview Summary is generated by AI. Please verify accuracy by reading the full article.
A study in Aging found higher blood levels of theobromine, found in dark chocolate and coffee, linked to slower cellular aging. The study found an association, not a causal link, and didn't specify consumption amounts. Researchers noted theobromine's potential impact on gene activity and aging. The study's limitations include lack of dietary data and single-time-point analysis. Read the full article for more on: The role of epigenetic clocks in estimating biological age. Potential synergistic effects of theobromine with other chocolate components. Limitations of the study and what they mean for future research.

What Is the Frame of Discourse?

A crucial design choice in DEMOCRITUS is the definition of the *frame of discourse* within which causal claims are generated and evaluated. This frame is not the input article alone. Instead, the article serves as a seed for constructing a broader causal neighborhood consisting of related topics and claims that surround the article's subject matter. Concretely, DEMOCRITUS first performs topic discovery over the input document and then expands these topics using a language model to form a local discourse manifold. Causal claims are extracted and evaluated relative to this expanded topic set. As a result, the system reasons not only about what the article explicitly states, but about how those statements are situated within a larger landscape of causal discourse.

Example: Discourse Frame Induced by The Washington Post Chocolate Article

The following topics were automatically extracted and expanded from The Washington Post article on chocolate and aging, defining the causal discourse neighborhood used by DEMOCRITUS:

- anti-inflammatory foods recommendations
- association versus causation in aging studies
- balancing sugar intake with chocolate consumption
- cancer risk reduction strategies
- cocoa content and nutritional value
- coffee as a source of theobromine
- cognitive biases and positivity
- dark chocolate ingredient criteria
- dark chocolate's health effects
- dietary impacts on health
- dietary recommendations for chocolate consumption
- DNA methylation as gene expression regulator
- doctor's wellness advice
- effects of theobromine on aging
- epigenetic clocks and aging measurement
- epigenetic clocks and aging rates
- epigenetic clocks and biological aging
- genetic and epigenetic aging factors

DEMOCRITUS uses this list of topics to produce a rather detailed causal report, evaluating thousands of local causal models. Interestingly, although The Washington Post AI Overview highlights theobromine as a candidate driver, DEMOCRITUS does not necessarily elevate it into Tier 1 claims. The highest-scoring local causal model linking dark chocolate intake to cholesterol regulation achieves a score of 7.533, reflecting a dense, well-connected neighborhood with multiple edges that are redundantly supported across the extracted discourse environment. By contrast, the theobromine-centered model scores substantially lower (3.988). Although theobromine is narratively salient in the article, its associated causal neighborhood is comparatively sparse and weakly redundant: fewer explicit edges recur across discourse neighborhoods, and agreement across local models is limited. As a result, the evaluator conservatively demotes the theobromine pathway, illustrating the distinction between narrative salience and causal support under model agreement.

DEMOCRITUS is not a document summarizer. While the original Washington Post article is framed around aging and theobromine, the highest-scoring local causal neighborhood discovered by DEMOCRITUS emphasizes

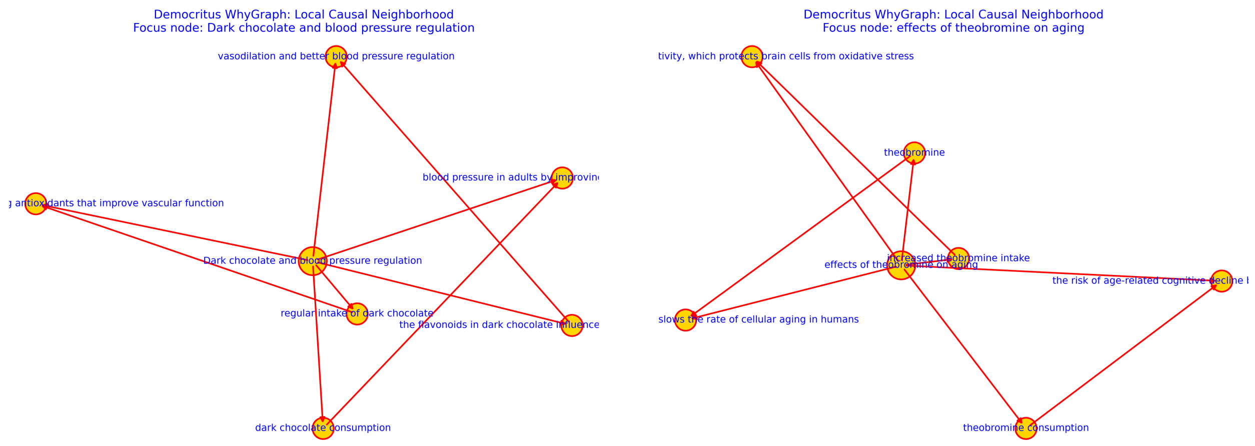

Democritus WhyGraph: Local Causal Neighborhood
Focus node: Dark chocolate and blood pressure regulation
vasodilation and better blood pressure regulation
blood pressure in adults by improving
g antioxidants that improve vascular function
Dark chocolate and blood pressure regulation
regular intake of dark chocolate
the flavonoids in dark chocolate influence
dark chocolate consumption
Democritus WhyGraph: Local Causal Neighborhood
Focus node: effects of theobromine on aging
tivity, which protects brain cells from oxidative stress
theobromine
increased theobromine intake
effects of theobromine on aging
the risk of age-related cognitive decline
slows the rate of cellular aging in humans
theobromine consumption

Figure 31: Contrasting local causal hypotheses for The Washington Post article on chocolate and aging case study. **Left:** a high-scoring local causal model (LCM) capturing a dense, redundant cardiometabolic neighborhood (e.g., dark chocolate intake $\rightarrow$ blood pressure/LDL/insulin sensitivity and related outcomes), which repeatedly appears across high-scoring hypotheses. **Right:** a lower-scoring LCM corresponding to the article's salient biochemical mechanism hypothesis involving theobromine (and/or methylation/epigenetic-clock pathways). The high-scoring model forms a connected, mechanism-like neighborhood (dark chocolate intake $\rightarrow$ flavonoids/antioxidants $\rightarrow$ vascular function/blood pressure) with multiple edges that are repeatedly supported by extracted discourse statements; the evaluator therefore accumulates evidence across several edges. In contrast, the theobromine model is comparatively sparse and fragmented: it contains fewer supported edges and weaker redundancy of edge-level support in the discourse sample. As a result, the theobromine pathway—while narratively salient in the article—does not survive as a top-ranked backbone hypothesis under model agreement.

cardiometabolic effects (e.g., cholesterol). This is not a failure of the system but an artifact of the evaluation objective: our default scorer ranks hypotheses by redundancy and model agreement in an expanded discourse environment, which favors dense and repeatedly expressed health-outcome mechanisms. To align the output with a user's question (e.g., aging), one can use a target-conditioned scoring variant that reweights hypotheses toward query-relevant variables while retaining the same adjoint reconstruction machinery. The selection of a scoring function is subjective: other choices may well change the ranking of causal models.

This example clarifies an important distinction between LLM summarization and DEMOCRITUS. A document summary naturally emphasizes salient narrative elements (here, theobromine). In contrast, DEMOCRITUS ranks causal hypotheses by redundancy and agreement across many local models. As a result, a dense cardiometabolic neighborhood can dominate the top-ranked models, while a mechanistically plausible but weakly supported pathway (theobromine $\rightarrow$ aging) appears only in lower-ranked hypotheses. This example illustrates why DEMOCRITUS is not a document summarizer. The evaluator rewards hypotheses whose edges are redundantly supported across the extracted discourse environment. The dark-chocolate/blood-pressure neighborhood contains multiple supported edges and therefore scores highly, while the theobromine neighborhood is sparse and less redundantly supported, yielding a lower score. Thus, DEMOCRITUS elevates what is structurally supported by model agreement rather than what is merely narratively salient in a single document. This example illustrates a key difference between narrative summarization and DEMOCRITUS-style causal analysis. Although The Washington Post article highlights theobromine as a salient candidate mechanism, the extracted discourse graph may contain too few explicit, repeated relations involving theobromine to form a connected local causal model.

By contrast, a cholesterol-centered neighborhood around dark chocolate intake contains multiple supported edges (e.g., dark chocolate intake reducing LDL cholesterol and influencing HDL cholesterol). Because the evaluation functional rewards redundant, evidence-supported edges and penalizes unsupported complexity, such dense mechanistic neighborhoods dominate the top-scoring models and therefore the Tier 1 backbone. This behavior is conservative: DEMOCRITUS elevates what is structurally supported by discourse redundancy rather than what is narratively salient.

Many deployed systems now provide AI-generated summaries of news articles and reports. For example, The Washington Post includes an AI-generated overview for its article on dark chocolate consumption and aging. That summary correctly highlights the main findings, notes that the reported relationship is associative rather than causal, and lists several caveats of the underlying study. Such summaries are useful as narrative condensations of text. However, they do not evaluate competing causal hypotheses, nor do they provide a ranked account of which causal claims are most robust under variation. They summarize *what is said*, but not *which causal claims are most credible* within the discourse.

In contrast, DEMOCRITUS produces a credibility report rather than a single summary. Starting from the same article, the system constructs thousands of local causal models, evaluates them numerically, and reconstructs a tiered ranking of causal claims based on agreement across high-scoring models. The resulting executive summary explicitly distinguishes between backbone claims shared across models and weaker, model-specific extensions. This distinction is crucial. DEMOCRITUS is not intended to replace LLM-based document summarization. Instead, it provides a complementary capability: a structured, model-grounded assessment of the causal claims expressed in text, together with explicit uncertainty and failure modes.

Why DEMOCRITUS *Works*

Evidence Expansion Protocol

DEMOCRITUS augments the input document with additional discourse evidence generated by a large language model. This expansion follows five principles:

1. **Hypothesis generation only:** Expanded statements are treated as candidate discourse, not as causal ground truth.
2. **Redundancy over authority:** Credibility arises from agreement across multiple paraphrases and local models, not from any single statement.
3. **Auditability:** All generated evidence is stored and can be inspected or ablated; downstream processing is deterministic.
4. **Conservative reconstruction:** The right adjoint suppresses claims that do not persist across high-scoring hypotheses.
5. **Fail-safe behavior:** In the presence of noise or inconsistency, the system yields low-confidence or empty summaries rather than spurious explanations.

At first glance, the behavior of DEMOCRITUS may appear surprising. Given only unstructured text, the system produces coherent, ranked causal narratives that often align with expert understanding. This section clarifies why this behavior is neither mystical nor accidental, but rather the consequence of several interacting design principles. A key reason DEMOCRITUS works in practice is that it does not rely solely on a single document. Instead, it exploits redundancy by constructing an expanded discourse environment around the input text. Modern large language models enable low-cost generation of supporting statements, paraphrases, mechanisms, and alternative framings that increase overlap among local discourse neighborhoods. This redundancy is crucial: it enables agreement under variation, which in turn makes model scoring and right-adjoint reconstruction meaningful. In effect, the system evaluates causal discourse not from a single narrative thread, but from a locally redundant ensemble of related causal statements.

REDUNDANCY IN CAUSAL DISCOURSE. Natural language exhibits substantial redundancy when describing causal relationships. Important causal mechanisms are typically expressed multiple times, using different phrasings, examples, and levels of abstraction. DEMOCRITUS does not trust any single statement; instead, it exploits this redundancy by sampling many local discourse neighborhoods and identifying relationships that persist across them. As a result, stable causal claims emerge from agreement under variation rather than from any individual extraction.

LOCALITY BEFORE GLOBALITY. Rather than attempting to construct a single global causal model, DEMOCRITUS operates locally. Local causal models are generated around specific discourse foci and topics, reflecting the fact that causal reasoning in text is inherently contextual. This locality dramatically reduces the combinatorial complexity of hypothesis generation while preserving semantic coherence. Global structure is recovered only indirectly, through aggregation of local models that agree.

OVER-GENERATION WITH DISCIPLINED EVALUATION. DEMOCRITUS deliberately over-generates candidate causal hypotheses. Most local causal models are incorrect, incomplete, or irrelevant. This is not a failure mode but a design choice: hypothesis generation is cheap, while evaluation is selective. A simple semantic scoring functional suppresses models that are internally inconsistent, weakly supported, or overly complex. The system therefore behaves as an ensemble method over causal hypotheses, retaining only those that score well under evidence-based criteria.

AGREEMENT RATHER THAN INFERENCE. Crucially, DEMOCRITUS does not attempt to infer a single "true" causal graph. Instead, it identifies causal claims that are supported across many high-scoring local models. The right adjoint aggregates model-level agreement into discourse-level credibility scores. Claims that survive across multiple plausible hypotheses are assigned high credibility, while model-specific or fragile claims are down-weighted. This emphasis on agreement explains the robustness of the resulting summaries.

SEPARATION OF GENERATION AND RECONSTRUCTION. The system cleanly separates hypothesis generation from semantic discipline. Domain-specific constraints, keyword anchoring, and focus de-duplication are applied during adjoint reconstruction rather than during model generation. This separation allows the left adjoint to explore the hypothesis space freely, while the right adjoint enforces relevance and interpretability. As demonstrated empirically, this design prevents premature pruning while still producing focused, human-readable outputs.

SAFE FAILURE MODES. When discourse quality is poor or hypotheses are internally incoherent, DEMOCRITUS does not hallucinate confidence. Instead, the adjoint reconstruction yields sparse or empty high-credibility tiers. Such outcomes correctly signal epistemic uncertainty rather than masking it with spurious explanations. This behavior was observed consistently in domains where the discourse generator was noisy, and contrasts sharply with unconstrained language-model summarization.

DETERMINISM AFTER DISCOURSE COMPILATION. Finally, once discourse triples are extracted, all downstream processing is deterministic. Local model generation, evaluation, selection, and reconstruction depend only on

Domain	LCMs Generated	Top-K	Max Score	Tier 1 Size	Tier 2 Size	Compression
Antarctica	$\sim$ 3,000	5	6.6	6–8	4–6	$\sim$ 600:1
Dark Chocolate and Aging	$\sim$ 3,000	5	4.8	5–7	3–5	$\sim$ 600:1
Indus Valley	$\sim$ 2,700	5	3.5	0	4–6	$\sim$ 540:1
Dinosaur Extinction	$\sim$ 2,600	5	4.0	0–1	0–2	$\sim$ 520:1

Domain	Mean Score	Std. Dev.	Median	90th %ile	99th %ile	Max
Antarctica	$\approx$ 0.9	$\approx$ 1.4	0.0	$\approx$ 3.2	$\approx$ 5.8	6.6
Dark Chocolate and Aging	$\approx$ 0.6	$\approx$ 1.1	0.0	$\approx$ 2.4	$\approx$ 4.2	4.8
Indus Valley	$\approx$ 0.5	$\approx$ 0.9	0.0	$\approx$ 1.9	$\approx$ 3.0	3.5
Dinosaur Extinction	$\approx$ 0.4	$\approx$ 0.8	0.0	$\approx$ 1.6	$\approx$ 2.7	4.0

Figure 32: Top: Quantitative summary of DEMOCRITUS experiments across domains. From thousands of generated local causal models (LCMs), only the top-K models are retained for adjoint reconstruction, yielding compression factors exceeding 500:1. Tier sizes reflect the number of distinct causal claims identified at each credibility level. Mechanism-dominated domains (Antarctica, Dinosaur with API model) exhibit stable Tier 1 backbones, while lifestyle and socio-historical domains show greater explanatory diversity or uncertainty. Bottom: Score distribution statistics for local causal models across domains. Distributions are consistently heavy-tailed: the median score is near zero, while a small fraction of models occupy a high-scoring tail. Higher-quality discourse generators (API model) yield greater score concentration and higher extrema, while lower-quality generators (OSS) produce flatter distributions. This behavior supports interpreting the evaluator as a semantic energy function over causal models.

the fixed discourse artifact. This property enables reproducibility, ablation studies, and principled analysis of system behavior independent of the language model used for compilation.

Experimental Results

We evaluate the proposed adjoint framework between *causal discourse* and *causal models* through a series of end-to-end experiments across heterogeneous domains. Each experiment instantiates the same conceptual pipeline:

1. **Discourse compilation:** An LLM compiles an input document into a finite set of relational causal statements, stored as a JSONL file of subject–relation–object triples.
2. **Local model generation:** Thousands of local causal models (LCMs) are generated as neighborhood subgraphs over the induced discourse manifold.
3. **Model evaluation:** Each LCM is assigned a scalar plausibility score using a text-evidence–based semantic evaluator.

All downstream stages are deterministic given the extracted discourse triples. As a result, reproducibility reduces to preserving a single artifact: the discourse sample itself.

Quantitative Evaluation

We complement the qualitative analyses above with quantitative measurements that characterize the behavior of the discourse–model adjunction across domains. The primary quantities of interest are the size of the hypothesis space explored, the concentration of evaluation scores, and the compression achieved by the adjoint reconstruction. The table summarizes the scale, score concentration, and compression achieved across domains. In all cases, thousands of candidate models are reduced to a small number of high-credibility causal claims via the adjoint reconstruction.

Domains and Data

We report results here on four qualitatively distinct domains.

- **Climate science (Antarctica):** A New York Times article on Antarctic ice melt and global sea-level rise.
- **Health and lifestyle (The Washington Post Chocolate):** A Washington Post article on dark chocolate consumption and biological aging.
- **Historical socio-ecological systems (Indus Valley):** Discourse concerning environmental, agricultural, and urban dynamics of the Indus Valley civilization.
- **Mass extinction (Dinosaur Extinction):** Scientific summaries of the Chicxulub impact and the end-Cretaceous extinction.

Summary and Further Reading

We have presented DEMOCRITUS, which introduces a new paradigm for inferring causal content from documents. Given an input article, DEMOCRITUS can automatically construct an expanded frame of discourse, generate thousands of candidate causal models, evaluate the causal models and produce an executive summary. DEMOCRITUS goes beyond document summarization by LLMs. We describe experiments on additional domains, and illustrate the natural language causal summaries produced by DEMOCRITUS, along with a detailed discussion of related work in the Appendix, along with a summary of the various LLM models that were used to produce the expanded frame of discourse. We used four different LLM models, ranging in size from 80B to 235B parameters, and locally hosted models vs. remotely accessible models. DEMOCRITUS produces human-consumable executive summaries that can support scientific analysis, policy reasoning, and decision-making. This combination of formal structure and practical utility distinguishes the approach from both purely symbolic and purely end-to-end language-based systems. For a more detailed description of DEMOCRITUS, you can read the original Arxiv paper. [65]

[65] Sridhar Mahadevan. Large causal models from large language models, 2025d. URL https://arxiv.org/abs/2512.07796

One limitation of DEMOCRITUS currently is that it takes a significant amount of time to process an individual document, and scaling it to process a large collection is a challenge that will be addressed in future work. The causal models constructed currently are limited to simple DAG models, and in future, we plan to construct a richer space of models that allow feedback. The document topic extraction process can be improved, and allow user feedback. DEMOCRITUS has yet to be tested with respect to an objective evaluation metric, such as a comparative study to similar summaries produced by expert humans.

Topos Causal Models

In this chapter, we explore the universal properties underlying traditional causal inference by formulating it in terms of a *topos*. More concretely, we introduce topos causal models (TCMs), a strict generalization of the popular structural causal models (SCMs). [66] A topos category has several properties that make it attractive: a general theory for how to combine local functions that define "independent causal mechanisms" into a consistent global function building on the theory of sheaves in a topos; a generic way to define causal interventions using a subobject classifier in a topos category; and finally, an internal logical language for causal and counterfactual reasoning that emerges from the topos itself. A striking characteristic of subobject classifiers is that they induce an intuitionistic logic, whose semantics is based on the partially ordered lattice of subobjects. We show that the underlying subobject classifier for causal inference is not Boolean in general, but forms a Heyting algebra. We define the internal Mitchell-Bénabou language, a typed local set theory, associated with causal models, and its associated Kripke-Joyal intuitionistic semantics. We prove a universal property of TCM, namely that any causal functor mapping decomposable structure to probabilistic semantics factors uniquely through a TCM representation.

[66] Sridhar Mahadevan. Universal causal inference in a topos. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025f. URL https://openreview.net/forum?id=TOhpnECT10

Introduction

In recent years, there has been significant interest in categorical models of causality. Categorical approaches fundamentally differ from past work in causality in their focus on the elucidation of universal properties. In our previous work, we introduced the framework of *universal causality* based on the notion of universal properties in category theory: a causal property is universal if it can be defined in terms of an *initial* or *final* object in a category of causal diagrams, or in terms of a *causal representable functor* using the Yoneda Lemma. [67] For example, a structural causal model (SCM) is defined as a (deterministic) mapping from a collection of exogenous variables into a collection of endogenous variables, derived by "collating" local functions that serve as independent causal mechanisms. However, SCMs can be further analyzed in terms of their universal properties, such as categorical product, coproduct, limits and colimits, equalizers and coequalizers etc. These latter

[67] Sridhar Mahadevan. Universal causality. *Entropy*, 25(4):574, 2023. DOI: 10.3390/E25040574. URL https://doi.org/10.3390/e25040574

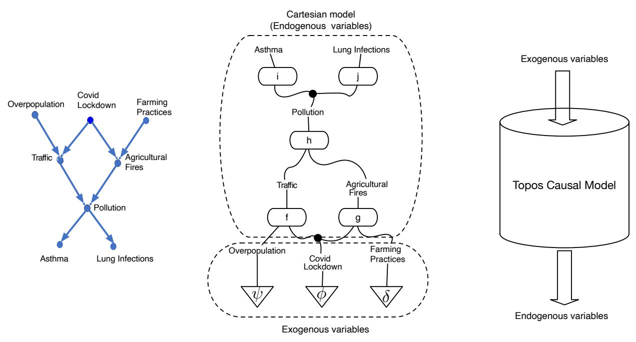


Figure 33: Topos causal models (TCMs) are defined as a category $\mathcal{C}_{\mathcal{TCM}}$ whose objects $c \in \mathcal{C}_{\mathcal{TCM}}$ are causal models, and whose arrows $\mathcal{C}_{\mathcal{TCM}}(c, c')$ are commutative diagrams between models c and c'. A specific object c defining a model can be conceptualized as a DAG (left, where information flows from top to bottom), or a string diagram in a Markov category (middle, where information flows from bottom to top), or in terms of its induced unique "blackbox" function mapping exogenous variables to endogenous variables (right).

properties can be shown formally to be initial or final objects in a category of diagrams, or as representable functors through the Yoneda Lemma.

Our main contribution in this chapter is to present a *topos-theoretic* view of causality, and in particular, introduce topos causal models (TCMs) that strictly generalize structural causal models (SCMs). A topos is a type of category, which is particularly well-suited to modeling operations that are "set-like". It also features an internal logical language. A topos provides three universal properties that make it natural as a category to do causal inference in: it provides a general theory for how to combine local functions, which can be viewed as "independent causal mechanisms", into a consistent global function building on the theory of sheaves in a topos. It enables a generic way to define causal interventions using a subobject classifier in a topos category. Finally, it gives an internal logical language for causal and counterfactual reasoning.

The objects in a TCM category can be conceptualized in multiple ways. First, each object can be a causal graphical model. Each object can also be a functor: for example, directed graphs form a topos functor category. TCMs can also be defined in terms of *string diagrams* in a symmetric monoidal *Markov category*, where we restrict ourselves to the Markov subcategory defined through deterministic morphisms. For example, the arrow h : Traffic $\otimes$ Agricultural Fires $\rightarrow$ Pollution defines a deterministic mapping specifying the two potential causes of Pollution. For exogenous variables, the arrow $\psi : I \rightarrow$ Overpopulation defines the marginal distribution on Overpopulation, where I is the terminal object in the Markov category. Finally, we can view a TCM object as a "blackbox" function that maps some collection of exogenous variables (e.g., "Overpopulation", or "Farming Practices" into some set of endogenous variables, e.g., "Asthma" or "Pollution").

Principles of Universal Causality

We give a brief overview of the fundamentals of universal causality (UC) before delving into the specific details of the TCM framework. UC rests on the Yoneda Lemma – any object in a category can be defined by the interactions it makes with other objects (upto isomorphism). In the setting of causal inference, it means that objects in a TCM category can be ascribed “meaning" through studying the arrows of the category, without having to “look inside" the object. The Yoneda Lemma states that the set of all morphisms into an object d in a category $\mathcal{C}$, sometimes denoted as $\mathbf{Hom}_{\mathcal{C}}(-,d)$, or as $\mathcal{C}(-,d)$, denoted as the *presheaf*, is sufficient to define d up to isomorphism. The category of all presheaves forms a *category of functors*, and is denoted $\hat{\mathcal{C}} = \mathbf{Set}^{\mathcal{C}^{op}}$. This category forms a topos, and will be fundamental to the TCM framework.

Lemma 4. Yoneda lemma*: For any functor $F : C \to \mathbf{Set}$, whose domain category $\mathcal{C}$ is “locally small" (meaning that the collection of morphisms between each pair of objects forms a set), and any object c in $\mathcal{C}$, there is a bijection $Hom(\mathcal{C}(-,c), F) \simeq Fc$ that associates 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.*

Definition 43. *A* **universal property** *of an object $c \in C$ in a category C is expressed by a representable functor F together with a universal element $x \in Fc$ that defines a natural isomorphism $C(-,c) \simeq F$. 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.*

We state two key results that underly UC. While both these results follow directly from basic theorems in category theory, their significance for causal inference is what makes them particularly noteworthy. The first result pertains to the notion of diagrams as functors, and shows that for the functor category of presheaves, which is a universal representation of causal inference, every presheaf object can be represented as a colimit of representables through the Yoneda Lemma. This result can be seen as a generalization of the very simple result in set theory that each set is a union of one element sets. The second result is the causal reproducing property, which shows that the set of all causal effects between two objects is computable from the presheaf functor objects defined by them. Both these results are abstract, and apply to any category representation of a causal model.

Theorem 9. Universality of Diagrams in UC*: In the functor category of presheaves $\mathbf{Set}^{\mathcal{C}^{op}}$, every functor object F is the colimit of a diagram of representable objects, in a canonical way.*

To explain the significance of this result for causal inference, note that UC represents causal diagrams as functors from an indexing category of diagrams to an actual causal model. The theorem above tells us that every presheaf object can be represented as a colimit of (simple) representable objects, namely functor objects of the form $\mathbf{Hom}_{\mathcal{C}}(-,c)$.

Theorem 10. Causal Reproducing Property: *All causal influences between any two objects c and d can be derived from its presheaf functor objects, namely*

$$\mathbf{Hom}_{\mathcal{C}}(c,d) \simeq \mathbf{Nat}(\mathbf{Hom}_{\mathcal{C}}(-,c), \mathbf{Hom}_{\mathcal{C}}(-,d))$$

Any causal influence of an object c upon any other object d can be represented as a natural transformation (a morphism) between two functor objects in the presheaf category $\hat{\mathcal{C}}$.

Topos Causal Models

Definition 44. *The category $\mathcal{C}_{\mathcal{TCM}}$ of topos causal models is defined as a collection of objects $c \in \mathcal{C}_{\mathcal{TCM}}$, each of which is a triple $\langle U, V, F \rangle$ where $V = \{V_1, \ldots, V_n\}$ is a set of* endogenous *variables, U is a set of* exogenous *variables, and F is a function from U to V. The arrows $\mathcal{C}_{\mathcal{TCM}}(c,d)$ are defined through commutative diagrams as illustrated below, where f and f' are the global functions induced by the TCM objects c and d, respectively.*

$$\begin{tikzcd}
U \arrow[r, "h"'] \arrow[d, "f"] & U' \arrow[d, "f'"'] \\
V \arrow[r, "g"] & V'
\end{tikzcd}$$

A submodel $c' = \langle U', V', F' \rangle$ of c is any subobject of c. The effect of an intervention on c is given by some submodel c'. Finally, let Y be a variable in V, and let X be a subset of V. The potential outcome in response to an intervention on X modeled by a submodel $c' \hookrightarrow c$ is the solution of Y in the submodel c'.

A commutative diagram, as the term suggests, is a structure showing the equivalence of two paths. Here, the diagram asserts that $g \circ f = f' \circ h$. In the context of our category $\mathcal{C}_{\mathcal{SCM}}$, the arrow $f : U \to V$ is simply an SCM M, and f is its induced mapping from exogenous to endogenous variables. Similarly, f' is also the induced function mapping exogenous to endogenous variables for another SCM M'. The morphisms h and g are functions on SCMs, which transform one causal model into another. In the specific case we are interested in, these functions define causal interventions, but in general, they may be arbitrary functions.

For completeness, we define a category $\mathcal{C}_{SCM}$ whose objects are indeed SCMs.

Definition 45. *The category $\mathcal{C}_{SCM}$ of structural causal models is defined as a collection of objects, each of which is a triple $\langle U, V, F \rangle$ where $V = \{V_1, \ldots, V_n\}$ is a set of* endogenous *variables, U is a set of* exogenous *variables, F is a set $\{f_1, \ldots, f_n\}$ of "local functions" $f_i : U \cup (V \setminus V_i) \to V_i$ whose composition induces a unique function F from U to V. Let X be a* *subset of variables in V, and x be a particular realization of X. A submodel $M_x = \langle U, V, F_x \rangle$ of M is the causal model $M_x = \langle U, V, F_x \rangle$, where $F_x = \{f_i : V_i \notin X\} \cup \{X = x\}$. The effect of an action $do(X = x)$ on M is given by the submodel M_x. Finally, let Y be a variable in V, and let X be a subset of V. The potential outcome of Y in response to an action $do(X = x)$, denoted $Y_x(u)$, is the solution of Y for the set of equations F_x.*

The set of arrows or morphisms between two objects c and d in the category $\mathcal{C}_{SCM}$, denoted $\mathcal{C}_{SCM}(c, d)$, represent ways of transitioning from SCM object c to d. For example, if d is a submodel of c, then the arrow defines a **do** calculus causal intervention.

Causal Inference in a Topos Category

We show in this section that a TCM category whose objects are defined as SCMs, and whose arrows correspond to commutative diagrams defining operations on causal models does define a topos. In the next section, we generalize from SCMs to consider more complex causal models over functor categories. Now, we can state the first key result of this paper.

Theorem 11. *The category $\mathcal{C}_{SC\mathcal{M}}$ forms a topos.*

Proof: Since we have previously defined the objects and arrows of the $\mathcal{C}_{SC\mathcal{M}}$ category, to show it forms a topos, we need to construct its subobject classifier. First, we need to define what a "subobject" is in the category $\mathcal{C}_{SC\mathcal{M}}$. Since SCMs can abstractly be defined as functions, let us assume that the SCM c that defines f is a *submodel* of the SCM c' that induces g. We can denote that by defining a commutative diagram as shown below. Let us stress the difference between the commutative diagram shown below Definition 44 for arbitrary functions g and h vs. the one below, where i and j are monic arrows.

$$\begin{tikzcd}
U \arrow[r, tail, "i"] \arrow[d, "f"'] & U' \arrow[d, "g"] \\
V \arrow[r, tail, "j"'] & V'
\end{tikzcd}$$

An element $x \in U'$, which is a particular realization of the exogenous variables in U', can be classified in three ways by defining a characteristic function ψ:

1. $x \in U$ – here we set $\psi(x) = \mathbf{1}$.
2. $x \notin U$ but $g(x) \in V$ – here we set $\psi(x) = \mathbf{\frac{1}{2}}$.
3. $x \notin U$ and $g(x) \notin V$ – we denote this by $\psi(x) = \mathbf{0}$.

The subobject classifier is illustrated as the bottom face of the cube shown on the right:

- $\mathbf{true}(0) = t'(0) = \mathbf{1}$
- $\mathbf{t} : \{\mathbf{0}, \frac{\mathbf{1}}{\mathbf{2}}, \mathbf{1}\} \rightarrow \{\mathbf{0}, \mathbf{1}\}$, where $\mathbf{t}(\mathbf{0}) = \mathbf{0}, \mathbf{t}(\mathbf{1}) = \mathbf{t}(\frac{\mathbf{1}}{\mathbf{2}}) = \mathbf{1}$.
- χ_V is the characteristic function of the exogenous variable set V.
- The base of the cube displays the subobject classifier $\mathbf{T} : \mathbf{1} \rightarrow \mathbf{\Omega}$, where $\mathbf{T} = \langle \mathbf{t}', \mathbf{true} \rangle$ that maps $\mathbf{1} = \mathbf{id}_{\{\mathbf{0}\}}$ to $\Omega = \mathbf{t} : \{0, \frac{1}{2}, 1\} \rightarrow \{0, 1\}$.

This proves that the subobject classifier for the category $\mathcal{C}_{\mathcal{SCM}}$ does not have Boolean semantics, but intuitionistic semantics as its subobject classifier Ω has multiple degrees of "truth", corresponding to the three types of classifications of monic arrows (in regular set theory, there are only two classifications). Moving on to show the other properties of a topos are satisfied, note that the terminal object is simply the identity function $\mathbf{id}_0 : \{0\} \rightarrow \{0\}$. Now, it remains to show that $\mathcal{C}_{\mathcal{SCM}}$ has pullbacks and exponential objects.

Pullbacks in $\mathcal{C}_{\mathcal{SCM}}$: Consider the cube shown on the left. Here, f, g, and h can be interpreted as three SCMs, each mapping some exogenous variables to some endogenous variables. The arrows i, j ensure that the bottom face of the cube is a commutative diagram, and the arrows p, q ensures the right face of the cube is a commutative diagram. The arrow from P to Q exists because looking at the front face of the cube, Q is the pullback of i and q, which must exist because we are in the category of **Sets**, which has all pullbacks. Similarly, the back face of the cube is a pullback of j and p, which is again a pullback in **Sets**. Summarizing, $\langle u, v \rangle$ and $\langle m, n \rangle$ are the pullbacks of $\langle i, j \rangle$ and $\langle p, q \rangle$.

Exponential objects in $\mathcal{C}_{\mathcal{SCM}}$: Now it only remains to check that the category has exponential objects. Let $f : U \rightarrow V$ and $g : U' \rightarrow V'$ be two functions induced by SCM models M and N. Then, we need to define the meaning of g^f in $\mathcal{C}_{\mathcal{SCM}}$, which we can define as $g^f : X \rightarrow Y$, where $Y = V'^V$, which must exist since **Sets** is a Cartesian closed category that has exponential objects (i.e., Y is simply the set of all functions from V to V'). Also, X is the set of all arrows in $\mathcal{C}_{\mathcal{SCM}}$ from SCM M to SCM N, which is the pair of functions $\langle h, k \rangle$ in the commutative diagram shown below. This finally proves that $\mathcal{C}_{\mathcal{SCM}}$ is a topos. □

Causal Models Over a Topos of Sheaves

We now describe a more general categorical framework for defining causal models as a topos by using the property that Yoneda embeddings of presheaves forms a topos. To ensure consistent extension into a unique global function, we build on the theory of sheaves, which ensures local functions can be "collated" together to yield a unique global function. In our setting, we will

$$\begin{tikzcd}
P \arrow[rr, "m"] \arrow[dr] \arrow[dd, "u"'] & & U'' \arrow[dr, "h"] \arrow[dd, "p"] & \\
& Q \arrow[rr, "n"] \arrow[dd, "v"] & & V'' \arrow[dd, "q"] \\
U \arrow[rr, "j"] \arrow[dr, "f"'] & & U' \arrow[dr, "g"] & \\
& V \arrow[rr, "i"'] & & V'
\end{tikzcd}
\qquad
\begin{tikzcd}
U \arrow[rr, "i"] \arrow[dr, "f"] \arrow[dd] & & U' \arrow[dr, "g"] \arrow[dd] & \\
& V \arrow[rr, "j"] \arrow[dd] & & V' \arrow[dd, "\chi_V"] \\
\{0\} \arrow[rr, "t'"] \arrow[dr, "id"'] & & \{0, 1/2, 1\} \arrow[dr, "t"] & \\
& \{0\} \arrow[rr, "true"'] & & \{0, 1\}
\end{tikzcd}$$

Figure 34: Left: diagram showing that $\mathcal{C}_{\mathcal{SCM}}$ has pullbacks. Right: The subobject classifier Ω for the topos category $\mathcal{C}_{\mathcal{SCM}}$ is displayed on the bottom face of this cube.

construct sheaves from categories over causal models through the Yoneda embedding $よ(x) : \mathcal{C} \to \mathbf{Sets}^{\mathcal{C}^{op}}$ and impose a Grothendieck topology.

Grothendieck Topology on Sites

Definition 46. *A* **sieve** *for any object x in any (small) category $\mathcal{C}$ is a subobject of its Yoneda embedding $よ(x) = \mathcal{C}(-, x)$. If S is a sieve on x, and $h : y \to x$ is any arrow in category $\mathcal{C}$, then*

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

Definition 47. *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 $t_x = \{f | cod(f) = x\}$ is in $J(x)$.*
2. *If $S \in J(x)$ then $h^*(S) \in J(y)$ for any arrow $h : y \to x$.*
3. *If $S \in J(x)$ and R is any sieve on x, such that $h^*(R) \in J(y)$ for all $h : y \to x$, then $R \in J(C)$.*

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

Definition 48. *A* **site** *is defined as a pair $(\mathcal{C}, J)$ consisting of a small category $\mathcal{C}$ and a Grothendieck topology J on $\mathcal{C}$.*

Definition 49. *The* **subobject classifier** *Ω is defined on any topos $\mathbf{Sets}^{\mathcal{C}^{op}}$ as subobjects of the representable functors:*

$$\Omega(x) = \{S | S \text{ is a subobject of } \mathcal{C}(-, x)\}$$

and the morphism $\mathbf{true} : 1 \to \Omega$ is $\mathbf{true}(x) = x$ for any representable x.

Universal Property of TCM over Functor Categories

Causal models, like SCMs, must represent both decomposable structure and (probabilistic) semantics. To capture this richer structure, we define TCM over *functor categories*, where every object is a functor that maps structure to semantics. For example, the category of Bayesian networks can be modeled as a functor category from a Markov category to the category **FinStoch** of finite stochastic processes.

Theorem 12. *Given a causal functor $A : \mathcal{C} \to \mathcal{E}$, such as the Bayesian network functor F_{CDU}, from a small category $\mathcal{C}$ (e.g., a symmetric monoidal category such as a Markov category) to a cocomplete category $\mathcal{E}$ (e.g., the category* **Prob** *of probability spaces (see Theorem 14)), the functor R from $\mathcal{E}$ to presheaves, given by (where $c \in \mathcal{C}$ and $E \in \mathcal{E}$)*

$$R(E) : c \mapsto \mathbf{Hom}_{\mathcal{E}}(A(c), E)$$

has a left adjoint $L : \mathbf{Sets}^{\mathcal{C}^{op}} \to \mathcal{E}$ defined for each presheaf P in $\mathcal{C}^{op}$ as the colimit

$$L(P) = Colim\left(\int_{\mathcal{C}} P \xrightarrow{\pi_P} \mathcal{C} \xrightarrow{A} \mathcal{E}\right)$$

where $\int_{\mathcal{C}} P$ is the category of elements, whose objects are pairs (c, p), where c is an object of $\mathcal{C}$ and p is an element of $P(C)$ (recall P is a presheaf, i.e., a set-valued functor that maps each element c into a set), and its arrows are $(c', p') \to (c, p)$ for any morphism $f : c' \to c$ such that $pc = p'$.

Proof: Essentially, the theorem is stating that there is a pair of adjoint functors $L \vdash R$, defined as:

$$L : \mathbf{Sets}^{\mathcal{C}^{op}} \rightleftarrows \mathcal{E} : R$$

As defined earlier, a natural transformation between two functors $\tau : P \to R(E)$ is a family $\{\tau_c\}$ of maps indexed by the objects $c \in \mathcal{C}$, where each map τ_c is defined as the mapping:

$$\tau_c : P(C) \mapsto \mathbf{Hom}_{\mathcal{E}}(A(C), E)$$

which is natural in c. τ can also be defined as a set of arrows of $\mathcal{E}$ as $\{\tau_c(p) : A(c) \to E\}_{(c,p)}$ that is indexed by the objects (c, p) of the category $\int_{\mathcal{C}} P$ of elements of P. This fact implies that there is a bijection

$$Nat(P, R(E)) \simeq \mathbf{Hom}_{\mathcal{E}}(LP, E)$$

This bijection being natural in P and in E proves that L is a left adjoint functor to R. $\square$

Now, let us define a general causal functor as mapping from a decomposable symmetric monoidal category (e.g., a Markov category) to the symmetric monoidal category of probability spaces.

Definition 50. *A causal functor $F : \mathcal{C} \to$* **Prob** *maps from a general symmetric monoidal category $\mathcal{C}$ with a comonoidal "copy-delete" structure to the category of probability spaces* **Prob**, *where each object $(\Omega, \mathcal{F}, \mathbb{P})$ is a probability space, and the arrows are measure-preserving maps, namely $\mathbf{Prob}(c, d)$, where $c = (\Omega_c, \mathcal{F}_c, \mathbb{P}_c)$ and $d = (\Omega_d, \mathcal{F}_d, \mathbb{P}_d)$, where $f \in \mathbf{Prob}(c, d)$ is such that $\mathbb{P}_c(f^{-1}(A)) = \mathbb{P}_d(A)$ for all $A \in \mathcal{F}_d$.*

Theorem 13. *For each causal functor $A : \mathcal{C} \to \mathcal{E}$ from a small category $\mathcal{C}$ defining the structure of a causal model to a cocomplete category $\mathcal{E}$ defining its (probabilistic) semantics, there exists a colimit preserving functor $L : \mathbf{Sets}^{\mathcal{C}^{op}} \to \mathcal{E}$ such that $A = L \circ よ$, where よ is the Yoneda embedding.*

Proof: The proof is just a special case of a more general result in the theory of sheaves. To emphasize the importance of the co-completeness condition on $\mathcal{E}$, we use the following result that the category of probability spaces is co-complete. $\square$

Theorem 14. *The symmetric monoidal category* $\mathbf{Prob}$ *has all colimits of non-empty diagrams.*

Proof: The proof that $\mathbf{Prob}$ has coproducts and coequalizers is given in a recent PhD thesis.[68] Thus, we can choose $\mathbf{Prob}$ as our cocomplete category $\mathcal{E}$ $\square$

[68] Ruben van Belle. *Kan Extensions in Probability Theory*. PhD thesis, University of Edinburgh, 2024

We can finally state the two central results of our paper, the first (Theorem 15) establishes the universal property underlying TCMs, and the second (Theorem 16) shows that causal interventions define a Heyting algebra whose logic is intuitionistic.

Theorem 15. *Any causal functor $F : \mathcal{C} \to \mathcal{E}$ from a structural causal category $\mathcal{C}$ (such as a Markov category) to a semantic cocomplete category $\mathcal{E}$ (such as* $\mathbf{Prob}$*) factors uniquely through a TCM structure defined by the Yoneda embedding, as given in Theorem 13.*

Proof: The proof follows directly from the above results $\square$

Definition 51. *A* **Heyting algebra** *is a poset with all finite products and coproducts, which is Cartesian closed. That is, a Heyting algebra is a lattice, including bottom and top elements, denoted by* $\mathbf{0}$ *and* $\mathbf{1}$*, respectively, which associates to each pair of elements x and y an exponential y^x. The exponential is written $x \Rightarrow y$, and defined as an adjoint functor:*

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

In other words, $x \Rightarrow y$ is a least upper bound for all those elements z with $z \wedge x \leq y$. As a concrete example, for a topological space X the set of open sets $\mathcal{O}(X)$ is a Heyting algebra. The “law of the excluded middle", meaning $\neg x \vee x = \mathbf{true}$, does not always hold in a Heyting algebra.

Theorem 16. *For any TCM category defined as $\hat{\mathcal{C}} = \mathbf{Sets}^{\mathcal{C}^{op}}$ by the Yoneda embedding $よ(c)$ of a small causal category $\mathcal{C}$, the partially ordered set $\mathrm{Sub}_{\hat{\mathcal{C}}}(P)$ of subobjects generated by causal interventions on any causal functor defined by the presheaf P is a Heyting algebra.*

Proof: This result follows directly from the corresponding result for any category of presheaves, and is based on constructing the complete lattice $\mathrm{Sub}(P)$ of all subfunctions of P using a pointwise operation for each object $c \in \mathcal{C}$, which can be shown to satisfy an infinite distributive law. $\square$

Causal Mitchell-Bénabou Language and its Kripke-Joyal Semantics

The Causal Mitchell-Bénabou language (CMBL) is a typed local set theory whose syntax and semantics is defined using the arrows of the $\mathcal{C}_{\mathcal{TCM}}$ topos. The types of CMBL as causal model objects M of $\mathcal{C}_{\mathcal{TCM}}$. For each type M, we assume the existence of variables $x_M, y_M, \ldots$, where each such variable has as its interpretation the identity arrow $\mathbf{1} : M \to M$. We can construct product objects, such as $A \times B \times C$, where terms like σ that define arrows are given the interpretation $\sigma : A \times B \times C \to D$.

- Each variable x_M of type M is a term of type M, and its interpretation is the identity $x_M = \mathbf{1} : M \to M$, Here, M may represent an entire SCM, an individual variable, or a causal functor mapping a Markov category to the cocomplete **Prob** category.

- Terms σ and τ of types C and D that are interpreted as $\sigma : A \to C$ and $\tau : B \to D$ can be combined to yield a term $\langle \sigma, \tau \rangle$ of type $C \times D$, whose joint interpretation is given as $\langle \sigma p, \tau q \rangle : X \to C \times D$, where X has the required projections $p : X \to A$ and $q : X \to B$. A causal intervention modeled as an arrow $f : X \to Y$ in $\mathcal{C}_{TCM}$ can be composed with a term $\sigma : U \to X$ to yield a term of type Y as $f \circ \sigma : U \xrightarrow{\sigma} X \xrightarrow{f} Y$.

- Terms of type Ω are defined as *formulae* of CMBL and can be combined with the usual logical connectives $\wedge$, $\vee$, $\Rightarrow$, $\neg$ and quantifiers $\forall$, $\exists$ to obtain further terms again of type Ω. An expression such as $\forall x \ \psi(x, y)$ is interpreted by an arrow $Y \to \Omega$ (since x is not a free variable). A formula $\psi(x, y)$ in the topos $\mathcal{C}_{TCM}$ is defined to be *universally valid* in the topos if the corresponding arrow $\psi(x, y) : X \times Y \to \Omega$ factors through $\mathbf{true} : 1 \to \Omega$. A formula ψ without free variables is interpreted as an arrow $\psi : \mathbf{1} \to \Omega$ and is valid if it coincides with the arrow $\mathbf{true} : \mathbf{1} \to \Omega$.

- Indirect proofs (i.e., *reductio ad absurdum*) cannot be used in CMBL because the rule of the excluded middle $\psi \vee \neg\psi$ is not in general valid, nor is the axiom of choice generally true. Instead, the rules of intuitionistic predicate calculus need to be used.

- The Kripke-Joyal semantics for CMBL is specified using *generalized elements*. We define an element of a causal model by the morphism $x : \mathbf{1} \to M$. Thus, a generalized element $\alpha : N \to M$ represents the "stage of definition" of M by N. We specify the semantics of how an TCM model N supports any formula $\phi(\alpha)$, denoted by $N \Vdash \phi(\alpha)$ by $N \Vdash \phi(\alpha) \quad \text{if and only if} \quad \text{Im}\, \alpha \leq \{x | \phi(x)\}$. Stated in the form of a commutative diagram, this "forcing" relationship holds if and only if α factors through $\{x | \phi(x)\}$.

$$\begin{tikzcd}
 & \{x|\phi(x)\} \arrow[r] \arrow[d, tail] & \mathbf{1} \arrow[d, "\mathbf{true}"] \\
N \arrow[r, "\alpha"'] \arrow[ur, dashed] & M \arrow[r, "\phi(x)"] & \mathbf{\Omega}
\end{tikzcd}$$

Summary and Further Reading

In this chapter, we formally introduced Topos Causal Models (TCMs), which generalize the well-known Structural Causal Model (SCM) due to Pearl. TCMs are based on exploiting three fundamental properties of a topos (i) A subobject classifier, which is used to model causal interventions in a more general way than “graph surgery" (ii) An intutionistic logic where causal inference is modeled in a more powerful way than classical "true or false" Boolean logic. (iii) Finally, TCMs use the theory of sheaves to “glue" together local causal structures in a more elegant way than SCMs can.

In the next chapter, we will show a detailed experimental study of j-do-calculus (aka “judo calculus"), which uses TCMs to design a decentralized method for causal discovery. We will show the map-reduce sheaf-theoretic framework of TCMs allows a more scalable approach to discovery of causal models from data. To achieve this, we will need to generalize TCMs from using the Grothendieck topology on a category to using a more general Lawvere-Tierney topology $j : \Omega \to \Omega$ on the subobject classifier.

We briefly described the Mitchell-Bénabou language that provides the internal language of a topos, along with its Kripke-Joyal semantics. In the next chapter, we will describe it in more detail. A rigorous account of topos theory, along with the theory of sheaves, and its internal logic is given in this classic book. [69]

[69] Saunders Mac Lane and Ieke Moerdijk. *Sheaves in Geometry and Logic a First Introduction to Topos Theory*. Springer New York, New York, NY, 1992. ISBN 9781461209270 1461209277. URL http://link.springer.com/book/10.1007/978-1-4612-0927-0

Judo Calculus

In this chapter, we will introduce j-do-calculus, a powerful sheaf-theoretic framework for decentralized causal discovery (aka “judo" calculus). [70] Judo calculus is based on the Lawvere-Tierney topology defined on the subobject classifier in a topos.

[70] Sridhar Mahadevan. Intuitionistic j-do-calculus in topos causal models, 2025c. URL https://arxiv.org/abs/2510.17944; and Sridhar Mahadevan. Decentralized causal discovery using judo calculus, 2025b. URL https://arxiv.org/abs/2510.23942

JUDO CALCULUS is more formally defined as j-stable causal inference using j-do-calculus in a topos of sheaves $\mathbf{Sh}_J(\mathcal{C})$. Classical causality typically assumes a single, universal truth: either “X causes Y" everywhere or it does not (Boolean logic). However, in real-world applications – from biology to medicine and social science – causal effects depend on regime (age, country, dose, genotype, or lab protocol). Our proposed judo calculus formalizes this context dependence formally as local truth: a causal claim is proven true on a cover of regimes, not everywhere at once. The Lawvere-Tierney modal operator j chooses which regimes are relevant; j-stability means the claim holds constructively and consistently across that family. Judo calculus extends Pearl's do-calculus by requiring that interventions be stable along j-covers, and reduces to the classical case for the trivial topology.

From Classical Do-Calculus to j-Do-Calculus

In this paper, we describe how to generalize classical do-calculus to *j-stable causal inference* inside a topos of sheaves $\mathbf{Sh}_j(\mathcal{C})$, where regimes form a site $(\mathcal{C}, j)$ and observations/interventions are sheaves on that site.

We build on the framework of Topos Causal Models (TCM) introduced inthe previous chapter, where causal interventions are defined as subobjects. We generalize the original setting of TCM using the Lawvere-Tierney topology on a topos, defined by a modal operator j on the subobject classifier Ω. We introduce j-do-calculus, where we replace global truth with *local truth* (Kripke–Joyal semantics) and formalize causal interventions as *structure-preserving morphisms* that are stable along j-covers. j-do-calculus is a sound rule system whose premises and conclusions are formulas of the internal (intuitionistic) logic of $\mathbf{Sh}_j(\mathcal{C})$. We define j-stability for conditional independences and interventional claims as local truth in the internal logic of $\mathbf{Sh}_j(\mathcal{C})$. We give three inference rules that strictly generalize Pearl's insertion/deletion and action/observation exchange, and we prove soundness in the Kripke–Joyal semantics. We show how these rules specialize back to classical do-calculus when j is the trivial topology (Boolean case) and to regime-aware identification when j encodes experimental covers.

Classical Do-Calculus

We briefly review the notion of a structural causal model (SCM), and the classical notion of do-calculus. [71] Succinctly, any SCM M defines a unique function from exogenous variables to endogenous variables, and do-calculus models interventions as "sub-functions":

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

Definition 52. *A* **structural causal model** *(SCM) is defined as the triple $\langle U, V, F\rangle$ where $V = \{V_1, \ldots, V_n\}$ is a set of* endogenous *variables, U is a set of* exogenous *variables, F is a set $\{f_1, \ldots, f_n\}$ of "local functions" $f_i : U \cup (V \setminus V_i) \to V_i$ whose composition induces a unique function F from U to V.*

Definition 53. *Let $M = \langle U, V, F\rangle$ be a causal model defined as an SCM, and X be a subset of variables in V, and x be a particular realization of X. A* **submodel** *$M_x = \langle U, V, F_x\rangle$ of M is the causal model $M_x = \langle U, V, F_x\rangle$, where $F_x = \{f_i : V_i \notin X\} \cup \{X = x\}$.*

Definition 54. *Let M be an SCM, X be a set of variables in V, and x be a particular realization of X. The* **effect** *of an action $do(X = x)$ on M is given by the submodel M_x.*

Definition 55. *Let Y be a variable in V, and let X be a subset of V. The* **potential outcome** *of Y in response to an action $do(X = x)$, denoted $Y_x(u)$, is the solution of Y for the set of equations F_x.*

We can also model counterfactuals in the following way:

Definition 56. *Let Y be a variable in V and let X be a subset of V. The counterfactual sentence "The value that Y would have obtained had X been set to x" is defined as the potential outcome $Y_x(u)$.*

Do-calculus provides three algebraic rules for manipulating interventional expressions of the form $P(Y \mid do(Z), X, W)$ based on conditional independence statements in a causal graph G. The notation used is as follows: $G_{\bar{X}}$ means delete all arrows into X (surgical intervention on X); $G_{\underline{Z}}$: delete all arrows out of Z; $Z(W)$ denotes the subset of Z that are not ancestors of any node in W in $G_{\bar{X}}$; and finally, $G_{\bar{X},\overline{Z(W)}}$ denotes the intervention that deletes arrows into those Z-nodes that are not ancestors of W.

1. **Rule 1 (Insertion/Deletion of Observations).** If $(Y \perp Z \mid X, W)_{G_{\bar{X}}}$, then
$$P(Y \mid \mathrm{do}(X), Z, W) = P(Y \mid \mathrm{do}(X), W).$$

2. **Rule 2 (Action/Observation Exchange).** If $(Y \perp Z \mid X, W)_{G_{\bar{X},\underline{Z}}}$, then

$$P\big(Y \mid \mathrm{do}(X), \mathrm{do}(Z), W\big) \;=\; P\big(Y \mid \mathrm{do}(X), Z, W\big).$$

3. **Rule 3 (Insertion/Deletion of Actions).** If $(Y \perp Z \mid X, W)_{G_{\bar{X},\overline{Z(W)}}}$, then

$$P\big(Y \mid \mathrm{do}(X), \mathrm{do}(Z), W\big) \;=\; P\big(Y \mid \mathrm{do}(X), W\big).$$

These rules form a sound and complete system for deriving identities between observational and interventional distributions using only the graphical structure of G.

j-do-Calculus: A Birds-Eye View

In moving from classical do-calculus to j-do-calculus, we transition from causal models over graphs to general categories, specifically toposes. The simplest way to understand this transition is to note that a category $\mathcal{C}$ whose objects are functions $f : A \to B$ over sets, and whose arrows are commutative diagrams between functions f and g, defined as $\mathcal{C}(f, g)$ defines a topos. This result, which was shown in detail for the case of SCMs in the previous chapter, shows that causal inference in SCMs and graphs is intrinsically topos-theoretic. One can expand this simple result to cover more cases. For example, the category of graphs $\mathcal{G}$ can be defined to consist of two objects v and e, and two non-identity arrows from v to e. Each graph then is defined as a presheaf $\mathbf{Sets}^{\mathcal{G}^{op}}$, a functor that maps the objects v and e to the set of edges E and vertices V of the actual graph, and that maps the two non-identity arrows between v and e to the initial and terminal vertex of each edge. More generally, any (small) category $\mathcal{C}$ can be converted into a topos through the Yoneda embedding $\mathcal{C} \to \mathbf{Sets}^{\mathcal{C}^{op}}$, defined as $c \mapsto \mathcal{C}(-, c)$, and called the presheaf. The category of presheafs forms a topos.

j-DO-CACULUS rules are summarized in the following figure, and will be explained in more detail in this chapter. At the core of j-do-calculus (aka “judo calculus), the original Grothendieck topology used in TCMs is generalized to use the Lawvere-Tierney j-operator on the subobject classifier Ω.

Causal Models Over a Topos of Sheaves

The categorical framework underlying Topos Causal Models (TCMs) introduced in the previous chapter defines causal inference in a topos by using the property that Yoneda embeddings of presheaves forms a topos. To ensure consistent extension into a unique global function, we build on the theory of sheaves, which ensures local functions can be “collated" together to yield a unique global function. In our setting, we will construct

Symbol	Type	Meaning / Typical usage
$\mathbf{C}$	category	Site of "regimes/contexts" (objects are stages; arrows are refinements).
j	L–T topology	Lawvere–Tierney topology on $\mathbf{Sets}^{\mathbf{C}^{\mathrm{op}}}$; enforces which sieves are "covering."
$\mathbf{Sh}_j(\mathbf{C})$	topos	Sheaves on $\mathbf{C}$ for j (the j-reflective subtopos).
a_j	functor	j-sheafification (left exact reflector $\mathbf{Sets}^{\mathbf{C}^{\mathrm{op}}} \to \mathbf{Sh}_j(\mathbf{C})$).
$\mathcal{U}$	object	A stage (or (object) $U \in \mathcal{C}$) in $\mathbf{C}$.
$\mathcal{S} = \{S_i \hookrightarrow \mathcal{U}\}$	family	A j-cover of $\mathcal{U}$ (local charts that jointly "see" $\mathcal{U}$).
$\Vdash_j$	relation	Internal forcing in $\mathbf{Sh}_j(\mathbf{C})$; $\mathcal{U} \Vdash_j \varphi$ reads "φ holds j-stably at $\mathcal{U}$."
$X \perp\!\!\!\perp Y \mid Z$	formula	Conditional independence assertion (CI).
$\mathsf{do}(x)$	term	Pearl's do-operator (surgical intervention) internalized in $\mathbf{Sh}_j(\mathbf{C})$.
$\mathcal{G}_{\overline{X}}$	graph	Mutilated graph with incoming edges to X cut (intervening on X).
$\mathcal{G}_{\underline{Z}}$	graph	Graph with outgoing edges from Z cut (treating Z as "measurement").
$\mathsf{P}(\cdot)$	object	Internal probability in $\mathbf{Sh}_j(\mathbf{C})$; e.g., $\mathsf{P}(y \mid \mathsf{do}(x), z, w)$.

Table 6: **Glossary of symbols and notation.** Informal reading: j specifies which families of local charts count as covers; $\mathcal{U} \Vdash_j \varphi$ means *every* chart in a j-cover of $\mathcal{U}$ validates φ, hence φ is forced globally at $\mathcal{U}$.

sheaves from categories over causal models through the Yoneda embedding $よ(x) : \mathcal{C} \to \mathbf{Sets}^{\mathcal{C}^{op}}$ and impose a Grothendieck topology. TCMs were originally defined over Grothendieck topologies on categories (known as sites), and we will generalize that formulation here to Lawvere-Tierney topologies.

Lawvere-Tierney Topologies on a Topos

In the previous chapter defining TCM, the category of sheaves defining $\mathcal{C}_{\mathcal{TCM}}$ (e.g., sheaves over a Markov category) was given a Grothendieck topology. A more elegant framework is to use the Lawvere-Tierney topology on the subobject classifier Ω.

Definition 57 (Lawvere–Tierney causal topology)**.** *Let $\mathcal{E}$ be an elementary topos with subobject classifier Ω and distribution monad $\mathsf{Dist}_{\mathcal{E}}$. A* causal topology *on $\mathcal{E}$ is a Lawvere–Tierney topology $j : \Omega \to \Omega$ satisfying:*

$$j(\top) = \top, \qquad j(p \wedge q) = j(p) \wedge j(q), \qquad j(j(p)) = j(p),$$

where $\top = \mathbf{true}$.

The original Grothendieck topology formulation is a special case of this more general formulation, which we expand on in greater depth in the remainder of the paper. In particular, we have the following result.

Theorem 17. *If $\mathcal{C}$ is a small category, the Grothendieck topologies J on C*

j-do rules at a glance

Figure 35: The Rules of j-do-calculus.

All equalities are identities *internal* to $\mathbf{Sh}_j(\mathbf{C})$ and read at stage $\mathcal{U}$ (i.e. under $\mathcal{U} \Vdash_j \cdots$). Each premise means: there exists a j-cover $\mathcal{S} = \{S_i \to \mathcal{U}\}_i$ such that the stated CI holds on *every* chart S_i after the indicated graph surgery.

[j-Rule 1: insert/delete observations]

$$\Big(Y \perp Z \mid X, W \text{ in } \mathcal{G}_{\overline{X}} \text{ on a } j\text{-cover of } \mathcal{U}\Big) \implies \mathsf{P}(y \mid \mathrm{do}(x), z, w) = \mathsf{P}(y \mid \mathrm{do}(x), w).$$

Reading: After cutting arrows into X, if every chart blocks Z from Y given X, W, then observing Z is irrelevant under $\mathrm{do}(x)$.

[j-Rule 2: action/observation exchange]

$$\Big(Y \perp Z \mid X, W \text{ in } \mathcal{G}_{\overline{X},\underline{Z}} \text{ on a } j\text{-cover of } \mathcal{U}\Big) \implies \mathsf{P}(y \mid \mathrm{do}(x), \mathrm{do}(z), w) = \mathsf{P}(y \mid \mathrm{do}(x), z, w).$$

Reading: After cutting arrows into X and *out of* Z, intervening on Z equals observing Z under $\mathrm{do}(x)$, chartwise.

[j-Rule 3: insert/delete actions]

$$\Big(Y \perp Z \mid X, W \text{ in } G_{\overline{X},\overline{Z(W)}} \text{ on a } j\text{-cover of } \mathcal{U}\Big) \implies \mathsf{P}(y \mid \mathrm{do}(x), \mathrm{do}(z), w) = \mathsf{P}(y \mid \mathrm{do}(x), w).$$

Reading: After cutting arrows into X and into the parents of Z not in W (i.e. $\overline{Z(W)}$), if every chart blocks Z from Y given X, W, then $\mathrm{do}(z)$ is irrelevant under $\mathrm{do}(x)$.

$$\begin{tikzcd}
\mathbf{Sh}(\mathcal{C}, J) \arrow[rr, dashed, "\simeq"] \arrow[dr, hook, "i_J"'] & & \mathcal{E}_j \arrow[dl, hook', "i_j"] \\
& {[\mathcal{C}^{\mathrm{op}}, \mathrm{Set}]} &
\end{tikzcd}$$

Figure 36: External Grothendieck topology J and internal Lawvere–Tierney topology j both induce subtopoi embedded in the presheaf topos $[\mathcal{C}^{\mathrm{op}}, \mathrm{Set}]$.

correspond exactly to Lawvere- Tierney topologies on the presheaf topos $\mathbf{Sets}^{\mathcal{C}^{op}}$.

The above figure gives a diagrammatic illustration of the relationship between the two approaches.

Kripke-Joyal Semantics for Sheaves

Every topos has an internal intuitionistic logic that derives from the fact that the subobject classifier Ω yields a poset of subobjects on which the semantics of a formal Mitchell-Bénabou language describing objects and arrows in the category can be defined. This formal language is associated with a Kripke-Joyal semantics, which we will specialize to a topos equipped with a Grothendieck topology, that is a site. This specialized structure captures how causal inference is woven in the fabric of the internal logic of a causal topos. Define $\mathtt{Sh}(\mathcal{C}, \mathcal{J})$ be a topos of sheaves with a specified Grothendieck topology $\mathcal{J}$, defined by the following diagram, where よ is the Yoneda embedding, and $\mathcal{P}$ is a presheaf:

$$\mathcal{C} \xrightarrow{よ} \mathcal{P}(\mathcal{C}) \xrightarrow{a} \mathbf{Sh}(\mathcal{C}, \mathcal{J}) \cong \mathcal{C}$$

where we know that the Yoneda embedding よ creates a full and faithful copy of the original category $\mathcal{C}$. Let us define the semantics for a sheaf element $\alpha \in X(C)$, where $X(C) = \mathbf{Sh}(\mathcal{C}, J)(\mathcal{C}(-, C), X))$. We will describe the Kripke-Joyal semantics in more detail later in the paper, but for now, a concise summary for the topos category of sheaves is as follows:

1. $C \Vdash \phi(\alpha) \wedge \psi(\alpha)$ if it holds that $C \Vdash \phi(\alpha)$ and $C \Vdash \psi(\alpha)$.

2. $C \Vdash \phi(\alpha) \vee \psi(\alpha)$ if there is a covering $\{f_i : C_i \to C\}$ such that for each i, either $C_i \Vdash \phi(\alpha)$ or $C_i \Vdash \psi(\alpha)$.

3. $C \Vdash \phi(\alpha) \to \psi(\alpha)$ if for all $f : D \to C$, and $D \Vdash \phi(\alpha \circ f)$, it holds that $D \Vdash \psi(\alpha \circ f)$.

4. $C \Vdash \neg\phi(\alpha)$ holds if for all arrows $f : D \to C$ in $\mathcal{C}$, if $D \Vdash \phi(\alpha \circ f)$ holds, then the empty family is a cover of D.

5. $C \Vdash \exists y\, \phi(x, y)$ holds if there is a covering $\{f_i : C_i \to C\}$ and elements $\beta_i \in Y(C_i)$ such that $C_i \Vdash \phi(\alpha \circ f_i, \beta_i)$ holds for each i.

6. Finally, for universal quantification, $C \Vdash \forall y\, \phi(x, y)$ holds if for all arrows $f : D \to C$ in the category $\mathcal{C}$, and all $\beta \in Y(D)$, it holds that $D \Vdash \phi(\alpha \circ f, \beta)$.

j-do-Calculus on Sites

To transition from classical do-calculus to j-do-calculus, we need to provide a "bridge" that maps from classical notions, like d-separation, to intuitionistic notions in j-do-calculus. We begin this transition by introducing some terms that will be used in the remainder of the paper.

STAGES AND GENERALIZED ELEMENTS. Let $(\mathcal{C}, J)$ be a site and $\mathbf{Sh}_J(\mathcal{C})$ its sheaf topos. For any object $A \in \mathbf{Ob}(\mathcal{C})$, a *generalized element of A at stage V* is a morphism $f : V \to A$ (equivalently, an element of the presheaf $yA(V) = \mathrm{Hom}_{\mathcal{C}}(V, A)$). The special case $\mathbf{1} \to A$ (where $\mathbf{1}$ is terminal) is a *global element*. In what follows we fix an *ambient context* (or *ambient object*) $U \in \mathbf{Ob}(\mathcal{C})$ and call any arrow $f : V \to U$ a *local stage over U*.

CHARTS ("REGIMES") AND J-COVERS. A *chart* (our earlier "regime") is precisely a local stage $f : V \to U$. A family of charts $\{f_i : V_i \to U\}_{i \in I}$

generates the sieve

$$\langle f_i \rangle = \{ h : W \to U \mid \exists i,\ \exists g : W \to V_i \text{ with } h = f_i \circ g \}.$$

We call $\{f_i\}$ a *J-cover of U* iff $\langle f_i \rangle \in J(U)$ (i.e. the generated sieve is J-covering).

READING FORMULAS "AT STAGE U". Let φ be a formula in the internal language. Write $U \Vdash_J \varphi$ to mean that φ is (internally) true at the ambient object U in $\mathbf{Sh}_J(\mathcal{C})$. In Kripke–Joyal semantics this is equivalent to the existence of a J-covering sieve $S \subseteq \mathrm{Hom}_{\mathcal{C}}(-, U)$ such that each local stage $f : V \to U$ in S *forces* φ after pullback:

$$U \Vdash_J \varphi \iff \exists\, S \in J(U) \text{ with } \forall f : V \to U \text{ in } S, \quad V \Vdash_J \varphi|_f.$$

Informally: *φ holds chartwise on a J-cover of U.*

GROTHENDIECK TOPOLOGY AND J-COVERS. A *sieve* S on U is *J-covering* iff $S \in J(U)$. We will say that a family of charts $\{f_i : V_i \to U\}$ is a *J-cover of U* iff the sieve it generates is J-covering:

$$\{f_i\} \text{ is a } J\text{-cover of } U \iff \langle f_i \rangle \in J(U).$$

Thus our earlier "J-cover" phrase always refers to a *covering family whose generated sieve is J-covering.*

LAWVERE–TIERNEY TOPOLOGY j AND J. The Grothendieck topology J on $\mathcal{C}$ corresponds to a Lawvere–Tierney topology $j : \Omega \to \Omega$ on the presheaf topos $\widehat{\mathcal{C}}$; the sheaf topos $\mathbf{Sh}_J(\mathcal{C})$ is the j-sheaf subtopos of $\widehat{\mathcal{C}}$. We freely pass between J (external/topological) and j (internal/logical) viewpoints; "j-closure" of a subobject corresponds to saturation under J-covering sieves. **Slogan.** A conditional independence $\varphi \equiv (X \perp\!\!\!\perp Y \mid Z)$ is *j-stable at a stage $\mathcal{U}$* iff the sieve of all refinements $u : \mathcal{V} \to \mathcal{U}$ that validate φ is a J-cover of $\mathcal{U}$.

SITE OF CAUSAL CONTEXTS. Fix a finite variable set $\mathcal{V}$ and a DAG G on $\mathcal{V}$. A *stage* is a pair $\mathcal{U} = (G, \sigma)$, where σ is a status profile that records which nodes are (i) conditioned/observed, (ii) intervened upon (incoming arrows cut), etc. A *morphism* $u : (G', \sigma') \to (G, \sigma)$ is a refinement that is identity on node names and *monotone in status* (a refinement may condition or intervene on more variables, but never less). Stages and refinements form a category $\mathbf{C}$.

OPEN PATHS AND SATISFACTION. For disjoint $X, Y, Z \subseteq \mathcal{V}$ and a stage $\mathcal{U} = (G, \sigma)$, let $\mathsf{OpenPaths}_{\mathcal{U}}(X, Y \mid Z)$ be the set of G-paths from X to

Y that are *d-open* under the usual collider/non-collider rules, evaluated after applying the surgeries in σ (e.g., $\mathrm{do}(\cdot)$). Write

$$\mathcal{U} \models (X \perp\!\!\!\perp Y \mid Z) \iff \mathsf{OpenPaths}_{\mathcal{U}}(X, Y \mid Z) = \varnothing.$$

THE SIEVE SELECTED BY A CI FORMULA. Given $\varphi \equiv (X \perp\!\!\!\perp Y \mid Z)$ and $\mathcal{U}$, define

$$\mathsf{S}_{\varphi}(\mathcal{U}) \ := \ \{\, u : V \to \mathcal{U} \text{ in } \mathbf{C} \ \mid \ V \models \varphi \,\}.$$

Lemma (sieve). $\mathsf{S}_{\varphi}(\mathcal{U})$ is a sieve on $\mathcal{U}$ (i.e., closed under precomposition).

Proof sketch. If $u : V \to \mathcal{U}$ validates φ and $w : W \to V$ is any arrow, then W refines V monotonically in status, which can only block additional paths; hence $W \models \varphi$ and $u \circ w \in \mathsf{S}_{\varphi}(\mathcal{U})$. □

GROTHENDIECK TOPOLOGIES FROM ADMISSIBLE CHARTS. Fix for each $\mathcal{U}$ a family $\{\rho_k : V_k \to \mathcal{U}\}_{k \in K}$ of *admissible local views* (charts) used to test CI at $\mathcal{U}$ (e.g., purely observational; or a mix including certain $\mathrm{do}(\cdot)$-surgeries). Let J be the Grothendieck topology *generated* by these bases: a sieve S covers $\mathcal{U}$ iff it contains a jointly epimorphic family refining $\{\rho_k\}$. Two canonical choices:

- J_{id} (classical): basis $= \{\mathrm{id}_{\mathcal{U}}\}$.
- J_{mix}: basis includes observational charts and specific interventional charts.

FORCING SEMANTICS (j-STABILITY). Write

$$\mathcal{U} \Vdash_J (X \perp\!\!\!\perp Y \mid Z) \quad :\iff \quad \mathsf{S}_{\varphi}(\mathcal{U}) \text{ is a } J\text{-cover of } \mathcal{U}.$$

Proposition (conservativity). With J_{id},

$$\mathcal{U} \Vdash_{J_{\mathrm{id}}} (X \perp\!\!\!\perp Y \mid Z) \iff \mathcal{U} \models (X \perp\!\!\!\perp Y \mid Z).$$

Reason. A sieve covers $\mathcal{U}$ in J_{id} iff it contains $\mathrm{id}_{\mathcal{U}}$. Thus $\mathsf{S}_{\varphi}(\mathcal{U})$ covers iff $\mathrm{id}_{\mathcal{U}} \in \mathsf{S}_{\varphi}(\mathcal{U})$, i.e., $\mathcal{U} \models \varphi$. □

Proposition (soundness of *j*-stability). If $\{\rho_k : V_k \to \mathcal{U}\}$ generates J at $\mathcal{U}$ and $V_k \models (X \perp\!\!\!\perp Y \mid Z)$ for all k, then $\mathcal{U} \Vdash_J (X \perp\!\!\!\perp Y \mid Z)$.

Reason. Each generator ρ_k lies in $\mathsf{S}_{\varphi}(\mathcal{U})$; hence the sieve they generate covers, and by upward closure of covering sieves, so does $\mathsf{S}_{\varphi}(\mathcal{U})$. □

WORKED MAPPING: EARTHQUAKE EXAMPLE. Let $\mathcal{U} = (G, \sigma)$ with $B \to A \leftarrow E$ and $A \to C$. Take J_{mix} generated by two charts: an *observational* chart ρ_{obs} (no conditioning on colliders unless stated) and an *interventional* chart $\rho_{\mathrm{do}A}$ that cuts the incoming edges into A. Then:

(i) $\mathcal{U} \Vdash_{J_{\mathrm{mix}}} (B \perp\!\!\!\perp E)$ (collider closed in obs; parents cut under $\mathrm{do}(A)$).
(ii) $\mathcal{U} \Vdash_{J_{\mathrm{mix}}} (B \perp\!\!\!\perp C \mid A)$ (chain blocked by A in both charts).
(iii) $\mathcal{U} \nVdash_{J_{\mathrm{mix}}} (B \perp\!\!\!\perp E \mid A)$ (conditioning on the collider opens the path in the obs chart).

TAKEAWAY. A CI formula φ determines a sieve S_φ; a Grothendieck topology J encodes which local views count as *covers*. Classical CI is truth at $\mathcal{U}$; j-stability is truth on a J-cover of $\mathcal{U}$—i.e., gluable from admissible local regimes.

CI AS AN INTERNAL PREDICATE. Fix a graph object G (DAG with surgery) represented in $\mathcal{C}$. For variables X, Y, Z (as objects/indices in G), let $\perp\!\!\!\perp_G\ (X; Y \mid Z)$ denote the internal formula "$X \perp Y \mid Z$ in G". Our usage

$$\text{“}Y \ \perp\!\!\!\perp\ Z \mid X, W \text{ in } \bar{G}^{(\cdot)} \text{ on a } J\text{-cover of } U\text{”}$$

means precisely: there exists a J-covering sieve $S \subseteq \mathrm{Hom}(-, U)$ such that for every $f\colon V \ \to\ U$ in S, the (pulled-back, surgically modified) graph satisfies $V \Vdash_J \perp\!\!\!\perp_G\ (X; Y \mid Z)$. By the clause above, this suffices to conclude $U \Vdash_J \perp\!\!\!\perp_G\ (X; Y \mid Z)$.

Algorithms for Judo Calculus

We describe an algorithmic and implementation framework for judo calculus, combining it with standard score-based, constraint-based, and gradient-based causal discovery methods. [72] We describe experimental results on how to (i) form data-driven j-covers (via regime/section constructions), (ii) compute chartwise conditional independences after graph surgeries, and (iii) glue them to certify the premises of the j-do rules in practice. We compare the regular and j-stable variants of popular causal discovery methods, from synthetic to real-world datasets from biology and economics. For more details, the reader is encouraged to refer to the original paper. [73]

[72] Alessio Zanga and Fabio Stella. A survey on causal discovery: Theory and practice, 2023. URL https://arxiv.org/abs/2305.10032

[73] Sridhar Mahadevan. Decentralized causal discovery using judo calculus, 2025b. URL https://arxiv.org/abs/2510.23942

We first give a high-level overview of judo calculus in this section, prior to introducing the highly technical definitions on which it is rigorously based. Judo calculus is *intuitionistic*, meaning that the law of the excluded middle $p \vee \neg p$ does not necessarily hold, and $\neg\neg p$ cannot be reduced to p, for any causal proposition p. Rather, judo calculus requires a constructive proof to determine a statement's truth. In the context of causality, this means you can't assume that a causal claim is either universally true or universally false without a proof for one or the other.

Judo calculus uses topos theory to model causality in a more flexible way than classical methods. For example, in many real-world applications of causal inference, a particular intervention, such as administering a drug or employing a lunch program in schools, may not be effective over the entire population, but rather in different "regimes" (e.g., senior citizens may respond more favorably than younger recipients, and similarly, students from low-income backgrounds may benefit nutritionally from a free lunch program than students from high-income regions). The following table summarizes the differences between classical do-calculus and judo calculus.

Characteristic	**Classical do-calculus**	**Judo Calculus**
Logic	Boolean: causal claims are globally true or false	Intuitionistic logic: truth is *local*
Context	Uses“average" treatment effect	Local truth is “glued" together
Interventions	“Surgery" of a causal graph	Subobject classifier
Identification	Axioms define three rules	More general axiomatic framework

Table 7: Some of the salient differences between classical do-calculus and judo calculus.

As a concrete example, let us imagine that a city planning a public health initiative to combat childhood obesity decides to distribute free healthy lunches in public schools. The city policy makers want to determine if this intervention reduces the students' body mass index (BMI) by the year's end. Classical do-calculus would seek a global average treatment effect over the entire school population. Judo calculus makes it possible to define individual regimes, such as “low-income" and “high-income" where the causal intervention may or may not be as effective.

Judo calculus works in the setting of sheaves or sites, which are categories that are equipped with a Lawvere-Tierney topology defined by a modal operator j on the subobjects. In plain English, this means that the category has a topology defined by the arrows, and their compositional structure. This j operator defines a notion of causal “stability", which will be studied extensively in the coming sections. For example, in the “low income" students, the city's causal intervention may be j-stable, whereas in the “high income" category, this intervention may not be as effective. The operator j acts on the subobject classifier Ω: an object in the category that serves to define “truth", which in general is not Boolean. The j operator acts to determine “local" truth from “global" truth, and also provides a closure property to determine j-stability. The j operator on Ω is defined to represent the notion of a “globally valid" causal statement. A j-cover represents a tesselation of the space of arrows that all have a common co-domain.

Judo Calculus Model of Causal Inference under Interference

In this section, we want to quickly show how our framework applies to the case when the standard Stable Treatment Unit Value Assumption (SUTVA) is violated because of “interference". For example, a report on a study involving power plants that may cause pollution among the residents of a particular county, where each individual resident may be exposed to the radiation from multiple power plants. A particular treatment on a power plant, such as imposing pollution controls, may affect many residents.

Classical do-calculus causal inference often assumes SUTVA (no interference): a unit's outcome depends only on its own treatment. In many spatial and networked problems this fails. Consider air pollution: a resident may be exposed to emissions from multiple plants, with exposure modulated by meteorology. Claims that ignore this heterogeneity are either false or too brittle to be useful. Our framework captures this by proving *local* causal truths on families of comparable regimes (*covers*) and then *gluing* them. We illustrate this in a two-source interference model with overlapping covers. Judo calculus shows how to certify claims locally on covers and glue them via j-stability. This simple example shows the power of the judo calculus formalism in modeling a range of real-world examples, and we will get into several real-world datasets in the experiments described later in the paper.

SETTING. Let $P = \{1, 2\}$ be two sources (plants) and R a population of residents. Time is indexed by t. Each plant has a binary treatment $Z_p(t) \in \{0, 1\}$ (e.g., a scrubber on/off). Meteorology $(\theta(t), M(t))$ denotes wind direction (degrees) and a mixing proxy (e.g., stability or mixing height). Interference arises because multiple sources contribute to each resident's exposure.

EXPOSURE MAPPING. For each resident r, define wind-dependent weights $w_{rp}(\theta, M) \geq 0$ (larger when r is downwind from p). Write the time-varying exposure as

$$E_r(t) \;=\; \sum_{p \in P} w_{rp}\big(\theta(t), M(t)\big)\, Z_p(t) \;+\; \eta_r(t),$$

and model the outcome as

$$Y_r(t) \;=\; \beta_1\, E_{r1}(t) \;+\; \beta_2\, E_{r2}(t) \;+\; \varepsilon_r(t),$$

where E_{rp} is the contribution traceable to source p. SUTVA is violated by design: Y_r depends on the vector $Z = (Z_1, Z_2)$ through the mapping $E_r(\cdot)$.

REGIMES AND COVERS. A *regime* is a subset of times where the exposure mapping is comparable (e.g., similar wind sector or mixing). Let

WL: west-large (230–310°); WS: west-small (250–290°); E: east (70–110°); LM: low mixing (e.g., $M < -0.5$).

These regimes *overlap*: we also consider intersections such as $WL \cap LM$ and $E \cap LM$. In practice, we may refine a regime into smaller *charts* (disjoint sub-regimes) to assess within-regime variability.

PRESHEAF VIEWPOINT. Let $\mathcal{R}$ be the (small) category whose objects are regimes $(E, WL, WS, LM, \ldots)$ and whose morphisms are inclusions (refinements). Define a presheaf $\mathcal{F} : \mathcal{R}^{op} \to \mathbf{Set}$ that assigns to each regime e the set of *local models or summaries* computable on e (e.g., regression coefficients, edge presence/absence), with restriction maps along inclusions. A *local truth* on e is a property $\varphi \in \mathcal{F}(e)$ we can build constructively from data in e. A *cover* $\mathcal{J} = \{e_i \to e\}$ is a jointly surjective family of inclusions; φ is *j-stable on e* if $\varphi|_{e_i}$ holds for every $e_i \in \mathcal{J}$ and these local sections are compatible (glue).

OPERATIONAL TEST FOR j-STABILITY (INTERFERENCE). We instantiate $\mathcal{F}$ with simple local regressions and a frequency threshold:

1. **Charts.** For each regime $e \in \{WL, WS, E, LM, WL \cap LM, E \cap LM\}$, split the time indices into K disjoint charts $e = \bigsqcup_{k=1}^{K} e^{(k)}$ (e.g., equal-size shards).

2. **Local models.** On each chart $e^{(k)}$, fit a standardized OLS model $Y \sim E_1 + E_2$. Record the coefficients $\widehat{\beta}_1^{(k)}(e), \widehat{\beta}_2^{(k)}(e)$. Declare the edge $E_1 \to Y$ *present* on the chart if $|\widehat{\beta}_1^{(k)}(e)| \geq \tau_\beta$ (e.g., $\tau_\beta = 0.2$); similarly for $E_2 \to Y$.

3. **Frequencies.** Compute edge frequencies $f_{E_1 \to Y}(e) = \frac{1}{K} \sum_k \mathbf{1}\{|\widehat{\beta}_1^{(k)}(e)| \geq \tau_\beta\}$, and $f_{E_2 \to Y}(e)$ analogously.

4. **Stability decision.** Fix a stability threshold $\pi \in [0, 1]$. We say $E_p \to Y$ is j-stable *on the cover* $\mathcal{J} = \{e_i \to e\}$ if $f_{E_p \to Y}(e_i) \geq \pi$ for all $e_i \in \mathcal{J}$. In words: the claim holds on *every chart* of *every member* of the cover.

The modality j is the Lawvere–Tierney topology that *closes* sieves under the chosen cover: truth is tested along the arrows $e_i \to e$.

WHAT WE OBSERVE. In a minimal simulation (two plants; wind-dependent weights), the west covers ($WL, WS, WL \cap LM$) yield $f_{E_1 \to Y} \approx 1.0$ and $f_{E_2 \to Y} \approx 0$; the east covers ($E, E \cap LM$) flip this, with $f_{E_2 \to Y} \approx 1.0$ and $f_{E_1 \to Y} \approx 0.5$–$0.6$. On LM, both edges are stable (e.g., $f_{E_1 \to Y} \approx 1.0$, $f_{E_2 \to Y} \approx 0.7$). Crucially, the same local claims persist on *intersections* (e.g., $WL \cap LM$, $E \cap LM$), illustrating that our sheaf-based proof obligation naturally handles overlapping charts.

GLUING INTO A STRUCTURE. When one aggregates many regimes, the local edges can be *glued* into a global π-stable skeleton by keeping edges whose frequencies exceed π in every member of the cover and on the intersections. In practice we select π by *validation likelihood* on held-out charts/regimes and then orient edges by a net-preference rule (e.g., keep $i \to j$ if $f_{i \to j} - f_{j \to i}$ exceeds a small margin), optionally imposing domain guards (e.g., forbid edges from the composite to its components).

WHY THIS MATTERS. Instead of a fragile global statement ("reducing emissions at plant 1 lowers hospitalizations everywhere"), we make a *j-stable* claim that is provably true on the relevant cover (e.g., west-wind regimes and their intersections with low mixing). This separation of concerns—*choose a cover, prove locally, glue globally*—is the essence of the j-do calculus: the usual identification rules apply, but only after claims have been made stable along the cover.

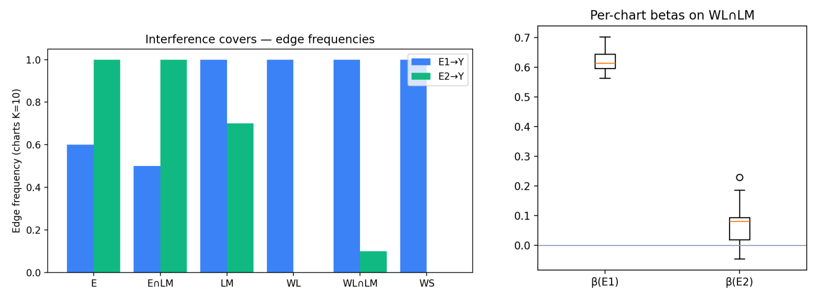


Figure 37: **Interference with overlapping covers.** Edge frequencies $f(E_1 \to Y), f(E_2 \to Y)$ by cover (left); per-chart coefficients on intersections (right). Local claims are j-stable on each cover and persist on intersections.

Computational and Statistical Efficiency of j-Stable Discovery

Another significant advantage of judo calculus is its highly decentralized characteristic. A J-cover $\mathcal{S} = \{ V_i \hookrightarrow U \}_{i=1}^{E}$ turns one hard pooled causal discovery problem into E *independent* subproblems plus a light-weight aggregation. This matches a map–reduce pattern:

$$\underbrace{\textsc{Discover}(U)}_{\text{pooled}} \rightsquigarrow \{ \underbrace{\textsc{Discover}(V_i)}_{\text{per-env/chart}} \}_{i=1}^{E} \text{ then } \underbrace{\textsc{Glue}(\{A_i\})}_{j\text{-aggregation}}.$$

In practice the map phase (per-env FCI/GES/DCDI) is *highly parallel*, and the reduce phase is an $O(Ep^2)$ Boolean fold over adjacency matrices, with short-circuit opportunities (below).

A SIMPLE COST MODEL. Let p be variables, n pooled samples, and n_i samples per chart ($\sum_i n_i = n$). For depth $d{=}0$ (marginal tests), Fisher-z CI tests are $O(n)$ each, and the skeleton step is $O(p^2 n)$.

$$T^{\text{pooled}}_{\text{FCI}}(n, p, 0) \approx c\, p^2 n,$$
$$T^{\text{sheaf}}_{\text{FCI}}(\{n_i\}, p, 0) \approx \sum_{i=1}^{E} c\, p^2 n_i = c\, p^2 n,$$

so the *sequential* work is comparable. However, with W workers the *wall-clock* time is

$$T^{\text{sheaf}}_{\text{wall}} \approx \max_{i \le E} c\, p^2 n_i + O(Ep^2),$$

often much smaller than $T^{\text{pooled}}_{\text{FCI}}$ because: (i) parallelism, (ii) better cache/memory behavior on smaller n_i, and (iii) short-circuiting during aggregation.

For $d{>}0$, the CI test count grows with the number and size of conditioning sets; the same parallel advantage holds, and smaller n_i usually makes regression/covariance subroutines cheaper and more numerically stable.

SHORT-CIRCUIT AGGREGATION. Let $A_i \in \{0,1\}^{p \times p}$ be per-chart adjacencies and Agg_τ denote a threshold rule "keep an edge if it appears in at least τ charts." Intersection and union are special cases with $\tau{=}E$ and $\tau{=}1$.

For intersection ($\tau{=}E$) a single zero in any chart kills the edge (stop early). For union ($\tau{=}1$) a single one confirms the edge. For k-of-E, both

d	Vanilla	J-stable	J-stable / Vanilla
10	30.3	27.8	**0.90**
20	51.8	42.3	**0.82**
40	143.5	101.9	**0.71**

Table 8: Comparison of regular (vanilla) vs. j-stable DCDI on a synthetic DAG benchmark.

early *accept* and *reject* bounds apply. This makes the reducer essentially linear in the number of *decisive* charts per edge.

MEMORY, I/O, AND PRIVACY. Per-chart runs touch only their own $n_i \times p$ block and emit a compact $p \times p$ binary matrix; the reducer never needs raw data. This also enables data-silo settings: share adjacencies, not samples.

STATISTICAL EFFICIENCY (STABILITY AS REGULARIZATION). Aggregating across charts is analogous to stability selection and ensemble model averaging:

- **Variance reduction.** Spurious edges that appear by chance in one environment are filtered by intersection or by a high support threshold.
- **Bias–variance tradeoff.** Intersection is conservative (low FP, higher FN); k-of-E balances sensitivity and specificity. The support curve (fraction of edges surviving threshold t) is an empirical stability diagnostic.
- **Implicit interventions.** Heterogeneity across charts (different regimes/perturbations) breaks symmetries that pooled data cannot, improving identifiability and power of CI tests.
- **Multiple testing control.** Viewing "edge present in chart i" as repeated evidence, support thresholds act as a robust filter against per-chart test noise without tuning α aggressively.

Experimental Validation of Judo Calculus Efficiency

To illustrate the gains with using the parallelism inherent in judo calculus, The table gives a comparison of the time required by the regular DCDI causal discovery method [74] with its j-stable variant. The figures below illustrate the computational benefits of decentralized judo calculus. With equal iteration budgets (10,000 iterations), j-stable's aggregation and π-selection adds negligible overhead: per-iteration wall-clock is on par with vanilla and is $\approx 10 - 30\%$ lower in our linear synthetic benchmarks as d, a parameter specifying the synthetic DAG, is varied. Because seeds/regimes are highly parallel, overall wall-clock for seed ensembles drops substantially with a few workers (here, set to 4).

[74] Philippe Brouillard, Sébastien Lachapelle, Alexandre Lacoste, Simon Lacoste-Julien, and Alexandre Drouin. Differentiable causal discovery from interventional data, 2020. URL `https://arxiv.org/abs/2007.01754`

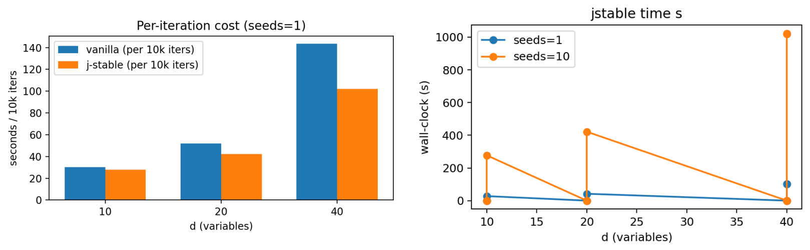


Figure 38: This figure illustrates the scalabilty of j-stable DCDI vs. regular DCDI.

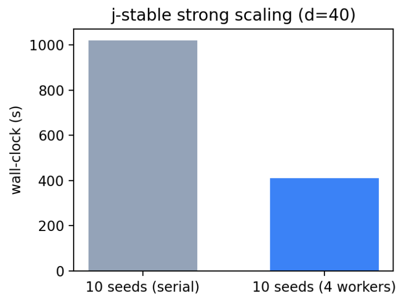


Figure 39: j-stable DCDI scales significantly better than regular DCDI on a synthetic DAG benchmark.

The j-stable do-operator (practical form)

Let J be a cover (the family of *comparable* regimes relevant for a query), and let $\{\mathbb{P}^e(\cdot)\}_{e \in J(w)}$ be regime-specific interventional models at covariate value w. We define the j-stable intervention probability by a *monotone* aggregator over the cover:

$$\mathbb{P}_J\big(Y \in A \mid \mathrm{do}_J(X{=}x), W{=}w\big) \quad := \quad \mathrm{Agg}_{e \in J(w)}\Big[\mathbb{P}^e\big(Y \in A \mid \mathrm{do}(X{=}x), W{=}w\big)\Big].$$

Semantics. Choose a cover J (e.g., countries with comparable measurement, QC-passed sites, or a wind sector in interference), evaluate the usual interventional conditional in each regime $e \in J(w)$, and combine them with a pre-registered, order-preserving Agg (Fisher/Stouffer pooling, trimmed mean, etc.). The result is a *local* do-query certified on J.

FROM STRUCTURE TO j-DO. Our discovery layer certifies structure on J via a stability map $F \in [0,1]^{d \times d}$: an edge $\{i, j\}$ is π-stable if $\max(F_{i \to j}, F_{j \to i}) \geq \pi$, and simple margins orient $i \to j$ when $F_{i \to j} - F_{j \to i} \geq \delta$. We select π (and sparsity for baselines) by validation likelihood on held-out regimes, and only then apply the rules of Judo calculus shown earlier.

Remark. The premises "$\cdot \mid (\cdot, J)$" are certified by the π-stable structure (and margins) on the cover. Formal details appear in the companion theory paper.

EXAMPLE (PISA ESCS). Let J be the set of OECD countries with comparable SES measurement; W include SES deciles; X = instruction time; Y = math score. Suppose discovery yields a π-stable skeleton/orientation

indicating that $Z{=}\text{ESCS}$ satisfies the j-backdoor premise for $X \to Y$ on J.
Then

$$\mathbb{P}_J\big(Y \le y \mid \text{do}_J(X{=}x)\big) \;=\; \text{Agg}_{e\in J} \sum_z \mathbb{P}^e\big(Y \le y \mid X{=}x, Z{=}z\big)\,\mathbb{P}^e(Z{=}z).$$

In words: evaluate the usual adjustment within each country and aggregate over the cover; the claim is local (on J) and certified by the π-stable structure.

Relation to transportability (Pearl–Bareinboim)

Transportability asks: given a source domain s (e.g., an RCT) and a target domain t, can we express $P_t(y \mid \text{do}(x))$ in terms of source quantities $P_s(\cdot)$ and target observables $P_t(\cdot)$? The method uses selection diagrams and classical do-calculus to derive adjustment/transport formulas; its claims are *global* within the chosen model and focus on identifiability of an effect under population shifts.

J-stability (judo calculus) asks: for a chosen family of comparable regimes (a cover) J, does a causal claim hold *locally and constructively* across the cover? Operationally, we certify a π-stable structure on J (frequency map F, π-skeleton, and net-preference orientation), and then evaluate J-do queries via an order-preserving aggregator:

$$\mathbb{P}_J\big(Y \in A \mid \text{do}_J(X{=}x)\,, W{=}w\big) := \text{Agg}_{e\in J(w)}\Big[\mathbb{P}^e\big(Y \in A \mid \text{do}(X{=}x), W{=}w\big)\Big].$$

HOW THEY CONNECT.

- *Transportability as a special case.* If the cover J contains the source and target ($\{s,t\} \subseteq J$) and we choose Agg to *select* the target component (the identity on t), then $\mathbb{P}_J\big(y \mid \text{do}_J(x)\big)$ reduces to $P_t(y \mid \text{do}(x))$. When a selection-diagram transport formula holds (e.g., backdoor on Z), the same Z is a J-admissible adjustment set and the J-backdoor rule yields the transported effect.

- *Broader locality.* Transportability typically assumes one global model with population differences mediated by effect modifiers. J-stability allows *covers with intersections* and context-dependent mechanisms: a claim may be certified on wheat-flour sites, winter/wind regimes, or genotype strata without committing to a single universal model.

- *Constructive robustness.* A transported formula says an effect *can* be mapped to t given assumptions. A J-stable claim says the effect *does* hold on the cover J because it is witnessed (e.g., by repeated per-regime fits) and glued by a monotone aggregator; the premises are empirically testable (stability margins, π-support).

TAKEAWAY. Transportability provides *formulas* for source$\rightarrow$target transfer; J-stability provides a *logic of local truth* and an empirical workflow (stability maps $\Rightarrow$ J-do) that subsumes transport as the two-regime, identity-aggregation case and extends it to richer covers where mechanisms and validity sets vary by context.
Example. Let $J = \{s = \text{RCT}, t = \text{registry}\}$ and suppose Z is J-admissible (stable backdoor). Then $\mathbb{P}_J(y \mid \mathrm{do}_J(x)) = \sum_z \mathbb{P}^t(y \mid x, z)\, \mathbb{P}^t(z) = P_t(y \mid \mathrm{do}(x))$, which matches the classical transport formula for $(s \to t)$.

Experimental Validation of j−Stable Causal Discovery

SETUP: REGIMES, PER-REGIME GRAPHS, AND AGGREGATION. Let $V = \{X_1, \ldots, X_d\}$ be the variables. We observe data partitioned into E *regimes* (a.k.a. environments) $\mathcal{E} = \{e_1, \ldots, e_E\}$ coming either from known labels (e.g. a column `env`) or from a preprocessing that induces regimes (e.g. clustering). For a causal learner $\mathcal{A} \in \{\text{GES}, \text{PSI-FCI}, \text{DCDI}\}$ and a regime $e \in \mathcal{E}$, we fit $\mathcal{A}$ on the subset $\mathsf{data}(e)$ and obtain a (possibly partially oriented) graph summary, represented here as an adjacency matrix

$$A^{(e)} \in \{0,1\}^{d \times d}, \qquad A^{(e)}_{ij} = 1 \iff \mathcal{A} \text{ asserts an edge } X_i \to X_j \text{ (or an adjaceny } X_i – X_j\text{) in regime } e.$$

From the E matrices we form the *support counts*

$$C_{ij} := \sum_{e \in \mathcal{E}} A^{(e)}_{ij} \in \{0, \ldots, E\}, \quad \text{and the stability ratios} \quad s_{ij} := \frac{C_{ij}}{E} \in [0,1].$$

These are the sufficient statistics of the *j-stable aggregation layer*. We report three derived graphs:

$$A^{\cap}_{ij} = \mathbf{1}[C_{ij} = E] \quad \text{(intersection)}, \qquad A^{\cup}_{ij} = \mathbf{1}[C_{ij} \geq 1] \quad \text{(union)}, \qquad A^{(k)}_{ij} = \mathbf{1}[C_{ij} \geq E - k] \quad \text{(“all-but-}k\text{”)}.$$

Optionally, a score threshold $\tau \in (0, 1]$ yields a soft variant $A^{(\tau)}_{ij} = \mathbf{1}[s_{ij} \geq \tau]$.

WHY THIS REALIZES j-STABILITY. In the theory, a conditional statement φ is *j-stable at stage* U iff there exists a J-cover $\{f_\alpha : V_\alpha \to U\}$ such that each chart V_α locally validates φ. Here, the regimes play the role of charts: each $e \in \mathcal{E}$ is a refinement of the ambient stage (same node set, refined observation/intervention status), and the learner $\mathcal{A}$ produces *local* edge claims $A^{(e)}_{ij} \in \{0, 1\}$. Intersection $A^{\cap}$ certifies edges that hold on *all* charts (cover-wise truth), hence correspond to *forced* edges under j. “All-but-k” $A^{(k)}$ is a robust variant: it requires truth on a cover after discarding at most k charts (useful when one regime is noisy or mis-specified). Union $A^{\cup}$ is purely diagnostic (upper bound on the skeleton).

LEARNERS WE EVALUATE.

- **GES** (score-based, Gaussian; we use BIC or Gaussian score).[75] Output is a CPDAG; we keep adjacencies for skeleton evaluation and, when available, directions for oriented metrics.
- **ψ-FCI** (constraint-based with selection bias handling).[76] We use Causal-Learn's FCI to get a PAG; we convert to an adjacency by treating any adjacency mark as 1 (for skeleton metrics) and $\to$ marks for orientation metrics when present.
- **DCDI** (gradient-based). Outputs a weighted adjacency; we threshold to obtain $A^{(e)}$.

[75] David Maxwell Chickering. Optimal structure identification with greedy search. *Journal of machine learning research*, 3 (Nov):507–554, 2002

[76] Amin Jaber, Murat Kocaoglu, Karthikeyan Shanmugam, and Elias Bareinboim. Causal discovery from soft interventions with unknown targets: Characterization and learning. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 9551–9561. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/6cd9313ed34ef58bad3fdd504355e72c-Paper.pdf

EVALUATION TARGETS AND METRICS. Let $A^{\text{true}} \in \{0,1\}^{d\times d}$ be the ground-truth adjacency of the synthetic DAG (directed). For any prediction A we report:

$$\text{TP} = \sum_{i\neq j} \mathbf{1}[A_{ij} = 1,\ A^{\text{true}}_{ij} = 1],\quad \text{FP, FN, TN analogously},\quad \text{precision} = \frac{\text{TP}}{\text{TP}+\text{FP}},\ \text{recall} = \frac{\text{TP}}{\text{TP}+\text{FN}},$$

$$\text{F1} = \frac{2\,\text{precision}\cdot\text{recall}}{\text{precision}+\text{recall}},\qquad \text{SHD} = \underbrace{\Delta_{\text{skeleton}}}_{\text{edge additions/deletions}} + \underbrace{\Delta_{\text{orientation}}}_{\text{arrow flips}}.$$

When a method produces only skeletons (e.g. PAG adjacencies), we compute SHD w.r.t. the true skeleton and skip orientation flips; we mark these rows as *undirected*.

EXPERIMENTAL PROTOCOL (ONE LINE PER METHOD).

1. **Per-regime fit.** For each $e \in \mathcal{E}$, run $\mathcal{A}$ on $\mathtt{data}(e)$ to get $A^{(e)}$; write `A_env_e*.csv`.
2. **j-stable aggregation.** Compute C, s, and one of $\{A^{\cap}, A^{(k)}, A^{(\tau)}\}$; write `A_Jstable_*.csv` plus `support_counts.csv` and `stability.csv`.
3. **Pooled baseline.** Run $\mathcal{A}$ on $\cup_e \mathtt{data}(e)$; write `A_pooled.csv`.
4. **Scoring.** Compare each prediction to A^{true} with the metrics above; for grid sweeps (e.g. α for FCI), tabulate best scores and produce a small LaTeX table.

WHAT THE AGGREGATION MEANS CAUSALLY. Edges in $A^{\cap}$ have *chartwise* support across the cover and thus behave like internally forced statements under the j-modality; they are stable under the refinements encoded by the regimes. Empirically, $A^{\cap}$ filters away edges that are brittle to regime changes (spurious under distribution shift), while $A^{(k)}$ retains edges that are stable except on at most k exceptional charts.

Inputs
synth_jsheaf/synth_data.csv $\to$ numeric vars $X_0,\ldots,X_{d-1}$ and categorical env $\in \{e0,e1,e2\}$ A_true_Jstable.csv $\to$ ground-truth directed adjacency

Step 1: per-env FCI (charts of a cover)
Split rows by env. For each $e \in \{e0,e1,e2\}$: run FCI on the numeric columns only $\Rightarrow$ PAG $\Rightarrow$ skeleton adjacency $A^{(e)} \in \{0,1\}^{d\times d}$.
Files: results_psifci_per_env/{ fci_enve0.csv, fci_enve1.csv, fci_enve2.csv }.

Step 2: *j*-stable aggregation across charts
Support counts: $C_{ij} = \sum_e A^{(e)}_{ij} \in \{0,1,\ldots,E\}$, stability ratio: $s_{ij} = C_{ij}/E$ with $E = 3$.
Intersection (forced on every chart): $A^{\cap}_{ij} = \mathbf{1}[C_{ij} = E]$.
All-but-k (robust gluing): $A^{(k)}_{ij} = \mathbf{1}[C_{ij} \geq E - k]$ (e.g., k=1).
Files: A_Jstable_fci.csv (intersection), support_counts.csv, stability.csv.

Step 3: pooled baseline (vanilla FCI)
Drop env and run one FCI on all rows $\Rightarrow$ pooled skeleton adjacency A^{pool}.
File: results_psifci_pooled/fci_envpooled.csv.

Step 4: evaluation vs. ground truth (undirected skeleton)
Compare $A^{\cap}$ (and $A^{(k)}$) and A^{pool} to A_true_Jstable.csv with undirected scoring: TP/FP/FN/TN, Precision, Recall, F_1, SHD.
Files: report.json in each output directory.

Figure 40: **Pipeline: PSI–FCI vs. *j*-stable PSI–FCI on synthetic data.** Regimes $e \in \{e0, e1, e2\}$ act as a *J-cover* $\{V_e \to U\}$ of the ambient stage U. Per-env FCI yields local skeletons $A^{(e)}$ (truth on each chart). Aggregation by intersection keeps the edges *forced on every chart* ($A^{\cap}_{ij} = 1 \iff A^{(e)}_{ij} = 1\,\forall e$), which operationalizes j-stability (local truth that *glues*). "All-but-k" allows up to k dissenting charts to trade precision for recall. The pooled baseline ignores regimes and may conflate heterogeneities.

DESIGN CHOICES (PRACTICAL KNOBS).

- *Depth/alpha (FCI).* We sweep $\alpha \in \{0.005, 0.01, 0.02\}$ at depth 0 for speed; depth > 0 is possible but slower.
- *Standardization.* For continuous data we z-score within each regime; pooled runs also drop non-numeric columns (e.g. `env`).
- *Thresholding (DCDI).* We set τ on learned weights to match the target edge budget or based on stability ratios s_{ij}.
- *Undirected vs directed scoring.* When orientation is unreliable (PAGs), we score skeletons and optionally report orientation flips where available (GES).

ABLATIONS AND SANITY CHECKS. We verify (i) SHD $= \Delta_{\text{skeleton}} + \Delta_{\text{orientation}}$ and (ii) $\Delta_{\text{dir\_sym}} = \Delta_{\text{skeleton}} + 2 \cdot \Delta_{\text{orientation}}$. We also report edge budgets $\sum_{i \neq j} A_{ij}$ for each run to make aggregation effects visible.

TAKEAWAY. The j-stable layer is a thin, method-agnostic "sheafification" of any base learner: compute per-regime graphs, then keep only edges that persist across a J-cover. This mirrors the theoretical slogan "local truth on a cover implies j-truth" and empirically improves precision under regime shifts while retaining recall when using an all-but-k aggregator.

STRUCTURE ACCURACY. We report SHD, F1 (skeleton and orientation), and orientation flip rate across regimes.

j-STABILITY INDEX. For edge logits (DCDI) or scores (TCES), define

$$\text{Stab} \;=\; 1 - \frac{1}{|E|} \sum_{(i,j)} \frac{\text{Var}_r[s_r(i \to j)]}{\text{Var}^{\max}_{r,e} + \varepsilon},$$

rescaled to $[0, 1]$. Higher is better (more stable).

OUT-OF-ENVIRONMENT GENERALIZATION. Hold out one or more regimes during training; report $\mathrm{NLL}_{\text{held-out}}$ for DCDI variants.

VIOLATION COUNT. Number of items flagged by the j-stability veto (lower is better).

WHAT TO REPORT (AND HOW TO REPRODUCE). We experimented with the following ablations:

1. **Runtime scaling.** Wall-clock vs. #charts E and workers W; compare pooled vs. j-stable (intersection and k-of-E).
2. **Support curve & Jaccard.** Plot $\#\{(u, v) : \text{support} \geq t\}$ and Jaccard with the union to visualize how strictness trades coverage for robustness.
3. **Accuracy (when GT available).** F1/SHD for pooled, intersection, and k-1 across α; include depth sweeps.
4. **Resource usage.** Peak RAM per job; note that per-chart jobs stay below pooled memory and are trivially parallelizable.

Method	Parallelism	Wall-clock	Robustness (FP control)
Pooled (single run)	none	high	low–moderate
j-stable (union)	map–reduce	low	low (liberal)
j-stable (k-of-E)	map–reduce	low	medium (tunable)
j-stable (intersection)	map–reduce	low	high (conservative)

Table 9: Compute/robustness trade-offs. All j-stable variants share the same cheap reducer; only the threshold changes.

TAKEAWAY. The sheaf viewpoint isn't just philosophically modular; it produces a *computational* modularity: run discovery locally, glue globally. This enables parallel speedups, early stopping in aggregation, better numerical behavior on smaller charts, and a principled stability mechanism that reduces false positives without expensive pooled runs.

Experimental Design

ASSUMPTIONS CHECKLIST.

- **Local consistency.** For each chart V_i, the chosen learner is consistent for $\Phi(G^\star|_{V_i})$. *Diagnostic:* report per-chart CI residuals / BIC deltas vs. learned $\widehat{G}_i$.
- **Separating cover.** The cover distinguishes non-edges and v-structures of interest. *Diagnostic:* list, for each true v-structure, a chart where the collider is not conditioned.

Method	Precision	Recall	F1	SHD
Pooled ψ-FCI	...	...	...	...
Per-env ψ-FCI (no gluing)	...	...	...	...
j-ψ-FCI (intersection)	...	...	...	...
j-ψ-FCI (all-but-k)	...	...	...	...

Table 10: Synthetic DAG: pooled vs per-env vs j-glued.

- **Interventional richness (if DAG uniqueness desired).** For each ambiguous node, there exists a chart that cuts all parents. *Diagnostic:* coverage table of "parents cut?" per node.
- **Overlap propagation.** Overlaps are large enough that Meek rules orient remaining edges. *Diagnostic:* orientation gain from overlap (before/after Meek closure).
- **Robustness.** If using all-but-k, assume exchangeability and bounded local error rates. *Diagnostic:* edge support histograms; separation of true vs. false edges.

REPORTING.

- **Edge support.** Histogram of per-edge support across charts, with thresholds used (intersection, all-but-k).
- **Metrics.** Precision/Recall/F1/SHD vs. ground truth per mode (pooled, per-env, j-intersection, j-all-but-k).
- **Ablations.** Vary k, cover size, and intervention availability; plot F1, SHD curves.

Experimental Setup

DATASETS. (i) Synthetic linear / nonlinear SCMs with $p \in \{10, 20\}$ variables, various densities and random interventions (perfect or soft); (ii) Sachs protein signaling (standard 11-variable benchmark) with interventional conditions.

REGIMES. We treat each intervention condition as a regime; observational data is one additional regime. Covers j include singletons and small unions (size 2–3), matching our aggregation granularity.

BASELINES. GES, CGES/TCES, FCI / ψFCI-TCM, DCDI / DCDI-TCM. All TCM variants share the same aggregator choices (mean or Fisher by default) and stability weight $\alpha \in \{0, 0.01, 0.05, 0.1\}$.

IMPLEMENTATION. For DCDI-TCM we enable `-tcm`, set `-tcm-alpha`, choose `-tcm-agg` (mean by default), and optionally `-ref-env` and `-jstability-veto`. Snapshots are written every K iterations for auditing.

REPORTING. Each condition averaged over 5 seeds; we provide SHD/F1, stability index, NLL on held-out regimes, and per-method runtime.

RELATION TO INVARIANCE. Our j-stability is compatible with invariance-based approaches (e.g., invariant causal prediction): invariance imposes equality of certain conditionals across environments; j-stability generalizes this as a sheaf-theoretic gluing condition on the algorithm's *decisions/statistics*, irrespective of the underlying parametric form, and integrates directly with search, constraints, or neural objectives.

Questions

Q1 Does j-stable aggregation improve structural accuracy over pooled baselines? **Q2** How sensitive are results to the CI threshold (α) and conditioning depth? **Q3** What is the computational impact of j-stability (wall-clock, parallelism)? **Q4** How do intersection vs. all-but-k (tolerance to one regime) trade FP/FN?

Datasets

Synthetic. Linear-Gaussian DAG with three regimes; ground-truth A_{true} known. **Sachs.** 11-protein phospho-signaling; we use standard ground truth. **LINCS (A375, 24h).** L1000 consensus signatures for cell line A375; regimes formed by dose binning / expression clustering; no ground truth (we report stability and overlap diagnostics).

Methods

Baselines. Pooled ψ-FCI and pooled GES (CausalLearn implementation, BIC score; z-score standardization). **j-stable variants.** Run the base learner independently in each regime $e \in E$, produce adjacencies $A^{(e)}$, then aggregate:

$$A_{\cap} = \bigwedge_{e \in E} (A^{(e)} > 0), \qquad A_{k\text{-allow}} = \mathbf{1}\left[\sum_{e} (A^{(e)} > 0) \geq |E| - k\right].$$

We report both *intersection* (k=0) and *all-but-1* (k=1).

Metrics

With ground truth: Precision, Recall, F1, Structural Hamming Distance (SHD). Without ground truth (LINCS): support histograms, Jaccard with pooled/union, and runtime.

Experimental protocol

Synthetic. Split by `env`; grid over $\alpha \in \{5\times10^{-4}, 10^{-3}, 2\times10^{-3}\}$ for ψ-FCI and depth $\in \{0,1\}$; GES uses BIC with standardization.
Sachs. `env` from k-means on standardized expression (k=3–6); evaluate pooled and j-stable ($\cap$, k=1).
LINCS. A375/24h subset; environments from dose quantiles or expression clustering (k=3); report stability and efficiency; pooled runs included when feasible.

Computational efficiency

Let $T_{\text{base}}(n,p)$ be the runtime of the base learner on n samples and p variables. Pooled runtime: $T_{\text{pooled}} \approx T_{\text{base}}(N,p)$. J-stable splits data into regimes E with sizes n_e and supports highly parallel execution:

$$T_{j\text{-stable}} \approx \max_{e\in E} T_{\text{base}}(n_e,p) \quad \text{(with } |E| \text{ workers)},$$

plus $O(Ep^2)$ aggregation. Empirically on A375 (p=30), per-env GES finished in $\mathcal{O}(10^2)$ seconds total with E=3, while pooled ψ-FCI was orders of magnitude slower or impractical. Besides speed, aggregation by $\cap$ (or k-allow) systematically reduces false positives—consistent with theoretical j-stability.

TAKEAWAYS. (i) On synthetic ground truth, j-stable aggregation substantially improves F1 and SHD for both GES and ψ-FCI at sensible α. (ii) On Sachs, trends are dataset/threshold dependent but k-allow offers a robust FP/FN trade. (iii) On LINCS, j-stable yields compact, high-support subgraphs and clear computational gains.

Experimental Results

In this section, we report on a diverse range of experimental domains, from synthetic to real-world, to give an illustration of the potential of judo calculus in causal discovery. A much wider set of experiments are currently in progress, and will be reported at a later date.

WHY j-STABLE ALGORITHMS OUTPERFORM THEIR COUNTERPARTS (INTUITION).

Pooled fits blur regime idiosyncrasies and let spurious partial correlations sneak in. j-stable fits each regime separately and keeps only edges that *repeat* across regimes. Under perfect interventions, true parents reappear consistently while spurious links do not. Aggregating into a frequency map

F and thresholding at π is a form of stability selection: the Binomial tail under the true edge rate p_T clears π with high probability, while the false edge rate p_F almost never does. A simple net-preference rule (keep $i \to j$ if $F_{i\to j} - F_{j\to i} \geq \delta$) resolves directions. Finally, selecting π/sparsity by validation log-likelihood on held-out regimes lands on a plateau where j-stable dominates vanilla—especially on denser graphs, where intersecting supports across regimes sheds many fragile edges.

Why j-stable discovery works: an ensemble view (bagging & boosting)

BAGGING IN DISGUISE. For each regime $e \in \mathcal{E}$ we fit a base learner $\mathcal{A}$ to get a directed graph $A^{(e)}$ (or a post-processed version of $W^{(e)}$). View the edge decision on e as a weak hypothesis $h_{ij}^{(e)} \in \{0,1\}$ for $i \to j$. We *bag* these hypotheses via the *frequency map*

$$F_{ij} \;=\; \frac{1}{|\mathcal{E}|}\sum_{e\in\mathcal{E}} h_{ij}^{(e)} \in [0,1], \qquad M_{ij} \;=\; F_{i\to j} - F_{j\to i} \in [-1,1] \;\;\text{(net margin).}$$

The π-stable skeleton keeps $\{i,j\}$ if $\max(F_{i\to j}, F_{j\to i}) \geq \pi$; we orient by a tiny margin rule $M_{ij} \geq \delta$ (with small $\delta > 0$), optionally with simple domain guards (e.g., forbid composite$\to$components).

WHY THIS HELPS. Across regimes, *true* edges tend to reappear (selection rate p_T), while *spurious* edges flicker (lower rate p_F). Over $E = |\mathcal{E}|$ regimes, the bagged vote for a true edge concentrates near p_T while a false edge concentrates near p_F; choosing π in the gap (p_F, p_T) is a *stability selection* filter:

$$\Pr\big[F_{ij} \geq \pi \;\big|\; \text{true}\big] \;\uparrow\; \text{fast in } E \quad \text{and} \quad \Pr\big[F_{ij} \geq \pi \;\big|\; \text{false}\big] \;\downarrow\; \text{fast in } E.$$

Perfect (or strong) interventions amplify this gap: when a node is targeted, its incoming edges *must* disappear in that regime and reappear elsewhere, creating a consistent on/off pattern that the vote captures. The same logic governs directions: the net margin M_{ij} for a truly directed edge drifts positive across regimes, while a symmetric or spurious pair hovers near zero.

WHY IT DOES NOT OVERFIT. We do not hand-pick π (or the per-seed sparsity K). Instead we *select* (K, π) by *validation log-likelihood* on held-out regimes (the same rule used in our synthetic and real-data experiments). This guards against the two failure modes—too permissive (many FPs) and too strict (recall collapse)—and lands on the broad plateau where j-stable outperforms vanilla.

A BOOSTING PERSPECTIVE (ONE PARAGRAPH). Bagging explains most of the gains, but the same picture admits a boosting-style extension: (1)

identify *where* the current ensemble underperforms (regimes/edges whose inclusion improves held-out regime LL but the vote disagrees), (2) upweight those (regime,edge) pairs, refit the base learner to produce a new per-regime component $\bar{A}^{(t)}$, and (3) add it to the ensemble with a stage weight chosen to maximize validation LL. This *J-Boost* idea—stage-wise corrections guided by held-out regimes—pushes stability up *where it helps prediction*, and leaves it alone elsewhere. We leave a full boosting study to future work; the bagging-style F-π aggregation already accounts for the large gains seen in our D10/D20 benchmarks.

Synthetic DAGs and data generation

We use linear-Gaussian SEMs with *perfect* single-node interventions across regimes. A graph is sampled once per instance, then we generate multi-regime data from that graph.

GRAPH SAMPLER (ACYCLIC BY CONSTRUCTION). Given number of variables d and an edge–density parameter e (average indegree),

1. Sample a random permutation π of $\{1, \ldots, d\}$ to define a topological order.

2. Set $m = \lfloor e \cdot d \rfloor$ and sample m ordered pairs (i, j) *uniformly* from the lower–triangular index set $\{(i, j) : i > j\}$ in the π-order. This yields a DAG adjacency $A \in \{0, 1\}^{d \times d}$ with $A_{ij} = 1$ iff $i \to j$.

3. Sample edge weights W_{ij} for $A_{ij} = 1$ i.i.d. as $S_{ij} \cdot U_{ij}$ with $S_{ij} \in \{\pm 1\}$ (equiprobable) and $U_{ij} \sim \text{Unif}[0.5, 2.0]$.

STRUCTURAL EQUATIONS (LINEAR-GAUSSIAN). With noise $\varepsilon_j \sim \mathcal{N}(0, 1)$ and the topological order,

$$X_j = \sum_{i \in \text{Pa}(j)} W_{ij} X_i + \varepsilon_j \quad \text{for } j = 1, \ldots, d.$$

We generate samples by forward substitution in the order π.

REGIMES AND PERFECT INTERVENTIONS. We consider R regimes $\mathcal{E} = \{e_0, \ldots, e_{R-1}\}$, with one observational regime e_0 and $(R-1)$ single–node interventions. In an interventional regime e that targets node $t(e)$ we *cut* all incoming edges into $t(e)$:

$$A^{(e)}_{it(e)} \leftarrow 0 \quad \text{and} \quad X_{t(e)} \leftarrow \mu_e + \varepsilon_{t(e)}$$

(where μ_e is an optional mean shift; we use $\mu_e = 0$ unless stated). All other equations remain unchanged. We draw n_{per} samples per regime (total $N = R \cdot n_{\text{per}}$), and we use the same DAG A across regimes.

Method	TP	FP	FN	TN	Precision	Recall	F1	SHD
PSI–FCI (pooled)	2	20	0	42	0.091	1.000	0.167	20
j-stable PSI–FCI (intersection)	1	5	1	57	0.167	0.500	0.250	6

Table 11: PSI–FCI vs. *j*-stable PSI–FCI at $\alpha = 0.005$ on the synthetic DAG.

	PSI–FCI (pooled)		*j*-stable (intersection)		*j*-stable (k=$E-1$)	
α	F1	SHD	F1	SHD	F1	SHD
0.005	0.286	10	0.333	**4**	0.200	8
0.010	0.286	10	0.333	**4**	0.200	8
0.020	0.267	11	0.333	**4**	0.200	8

Lower SHD is better. Flat lines across α indicate regime-wise skeletons were stable; the gain comes from cross-regime aggregation.

Table 12: **PSI–FCI vs. *j*-stable PSI–FCI on synthetic 3–regime data.** Scores are computed on the undirected skeleton (depth= 0). The *j*-stable *intersection* keeps an edge only if it appears in *all* regimes; $k = E - 1$ keeps edges present in at least $E-1$ regimes.

DEFAULT SETTINGS. Unless otherwise noted we use R=10 (one obs + nine single–node interventions), n_{per}=1000 (thus N=10,000), and seeds $\{123+g\}$ for graph index $g \in \{1, \dots, 10\}$. We report medians ± IQR over 10 graphs per condition. Evaluation uses directed and skeleton SHD against the known A.

The figure compares the vanilla ψ-FCI method with the *j*-stable ψ-FCI. For the specific value of $\alpha = 0.005$, the table shows that

- ψ-FCI (pooled): perfect recall but many false positives → low precision and larger SHD.
- j-stable ψ-FCI (intersection across envs): trades some recall for far fewer false positives → higher precision, higher F1, much smaller SHD.

The main takeaway at this setting is that enforcing j-stability improves overall accuracy (F1) and structure fidelity (lower SHD) by suppressing edges that aren't consistent across regimes. The pooled baseline is aggressive (recall=1.0) but overfits regime-specific artifacts (20 FPs).

The below figure shows the improvement in performance of the *j*-stable version of ψ-FCI against the (pooled) regular version for the synthetic DAG. To measure sensitivity to the CI level, we swept ψ-FCI's significance $\alpha \in \{0.005, 0.01, 0.02\}$ on the synthetic 3-regime dataset. Pooled ψ-FCI achieved $F1 \approx 0.27-0.29$ with SHD $10-11$. In contrast, the *j*-stable intersection aggregate (edge kept iff present in every regime) delivered $F1 \approx 0.33$ with SHD 4, i.e., fewer errors and better precision–recall balance. A more permissive j-stable rule $(k = E - 1)$ increased false positives and reduced F1 ($\approx$ 0.20, with SHD = 8). Across this α range the curves are flat, indicating the regime-wise PAG skeletons were already stable; the improvement comes from cross-regime invariance rather than per-env thresholding.

The figure compares the run-time efficiency of *j*-stable variant of ψ-FCI with the regular pooled version.

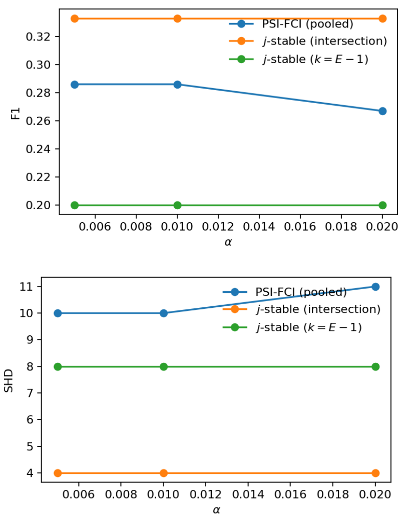


Figure 41: Comparison of regular ψ-FCI (pooled) with the j-stable variant on a synthetic DAG.

α	Method	F1	SHD	Time (s)
0.005	j-stable	0.25	6	1.799
0.005	pooled	0.167	20	1.065
0.01	j-stable	0.25	6	1.075
0.01	pooled	0.167	20	1.087
0.02	j-stable	0.25	6	1.093
0.02	pooled	0.154	22	1.085

Table 13: Comparison of j-stable vs. regular ψ-FCI on synthetic DAG.

Method	TP	FP	FN	TN	Precision	Recall	F1	SHD
GES (pooled)	6	14	0	61	0.30	1.00	0.462	14
j-stable GES (intersection)	6	0	0	75	1.00	1.00	1.00	0

Table 14: GES vs. j-stable GES on the synthetic DAG (undirected evaluation).

Synthetic DAG: GES vs. j-stable GES

We now turn to comparing the regular GES method vs. its j-stable variant, again on the synthetic DAG that was used in the previous section. The results are shown in the table below. The results from the table show that vanilla (pooled) GES gets full recall but lots of false positives, whereas j-stable GES (intersection across envs) removes those spurious edges and exactly matches the true undirected skeleton. These results are again consistent with what we saw above for the comparison of ψ-FCI against its j-stable variant.

DCDI synthetic setups (perfect interventions)

We now present results comparing the j-stable variant of DCDI against the “vanilla" DCDI method. The figure shows that the j-stable variant produces a much better performance, as measured by the SHD metric, with much lower variance. These experiments are on the “perfect" DAG benchmarks in DCDI. We replicate the linear settings with *perfect interventions* and no hidden confounding:

GRAPH SIZES AND DENSITIES. We use $d \in \{10, 20\}$ variables and average indegree $e \in \{1, 4\}$. Thus the expected #edges is $\approx e \cdot d$ (e.g., $d{=}10, e{=}1 \Rightarrow\sim$ 10 edges; $d{=}20, e{=}4 \Rightarrow\sim$ 80 edges). For each (d, e) we sample 10 independent DAGs using the sampler in §.

MECHANISM CLASS AND NOISE. Linear-Gaussian SEMs as above with i.i.d. $\varepsilon_j \sim \mathcal{N}(0, 1)$. (In the DCDI paper additional mechanism classes—ANM and nonlinear NN—are also reported; here we focus on the linear case.)

INTERVENTIONS AND SAMPLE BUDGET. We use R regimes comprising one observational regime and single–node perfect interventions (incoming edges to the target node are cut); targets are chosen uniformly without replacement until all nodes (or the budget) are covered. We use a total of $N{=}10{,}000$ samples per graph, *uniformly* distributed across regimes unless noted.

EVALUATION. We compute directed and skeleton SHD against the ground-truth DAG and report medians $\pm$ IQR over the 10 graphs per (d, e). For method selection we mirror the DCDI tuning rule: sparsity (and, for j-stable, the stability threshold π) is chosen by *validation log-likelihood* on held-out regimes; test LL is reported for the selected models.

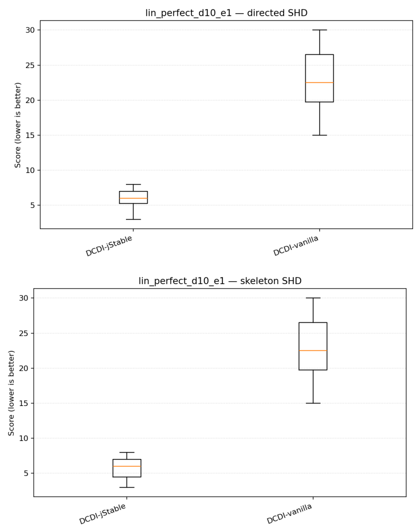


Figure 42: Results on Synthetic DAG comparing j-stable DCDI against the standard DCDI. Box-plots over 10 random graphs (whiskers 1.5×IQR). j-stable DCDI yields substantially lower SHD and tighter dispersion than vanilla DCDI in both directed and skeleton space (median 6.0 vs 22.5).

	Directed SHD		Skeleton SHD	
Condition	DCDI-vanilla	DCDI-jStable	DCDI-vanilla	DCDI-jStable
d=10, e=1	22.5 ± 6.75	$\mathbf{6.0 \pm 1.75}$	22.5 ± 6.75	$\mathbf{6.0 \pm 2.50}$
d=20, e=1	100.5 ± 7.00	$\mathbf{13.5 \pm 5.25}$	100.5 ± 7.0	$\mathbf{12.5 \pm 4.5}$
d=20, e=4	106.0 ± 8.25	$\mathbf{76.0 \pm 7.00}$	102.0 ± 2.75	$\mathbf{72.5 \pm 8.25}$

Table 15: Linear, perfect interventions: SHD (median ± IQR) over 10 random graphs. Lower is better.

The tables show the corresponding results for both the D10 and D20 synthetic dataset.

Sachs protein signaling (11 nodes, multiintervention)

We use the classical *Sachs* flowcytometry dataset (11 phosphoproteins/phospholipids, singlecell measurements) with multiple targeted perturbations. [77] The canonical variable set is

{raf, mek, erk, pka, pkc, pip2, pip3, plcg, akt, p38, jnk}.

[77] Karen Sachs, Omar Perez, Dana Pe'er, Douglas A. Lauffenburger, and Garry P. Nolan. Causal protein-signaling networks derived from multiparameter single-cell data. *Science*, 308(5721):523–529, 2005. DOI: 10.1126/science.1105809

REGIMES. Each stimulation/inhibitor setting defines a regime (e.g., `CD3CD28`, `PKA_inh`, `PKC_act`, ...). We create a label `env` from the condition name and treat regimes as independent "sites" for jstability.

PREPROCESSING. We concatenate the percondition singlecell tables, and drop the `env` column from the feature matrix; we logtransform/standardize features per run if not already standardized. We keep all regimes with sufficient sample size (default $\geq$200 cells).

DISCOVERY AND STABILITY. We run ten jstable DCDI seeds per ex-

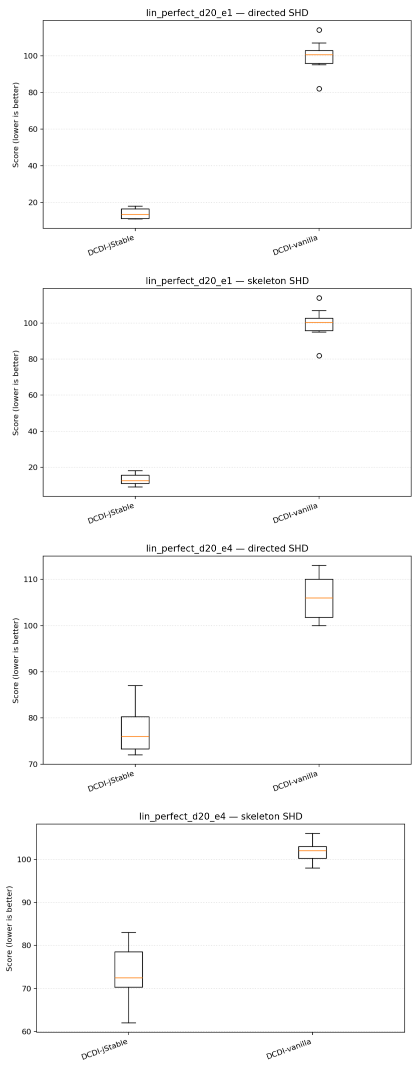


Figure 43: Results on Synthetic DAG comparing j-stable DCDI against the standard DCDI. Box-plots over 20 random graphs (whiskers 1.5×IQR). j-stable DCDI yields substantially lower SHD and tighter dispersion than vanilla DCDI in both directed and skeleton space (median 6.0 vs 22.5).

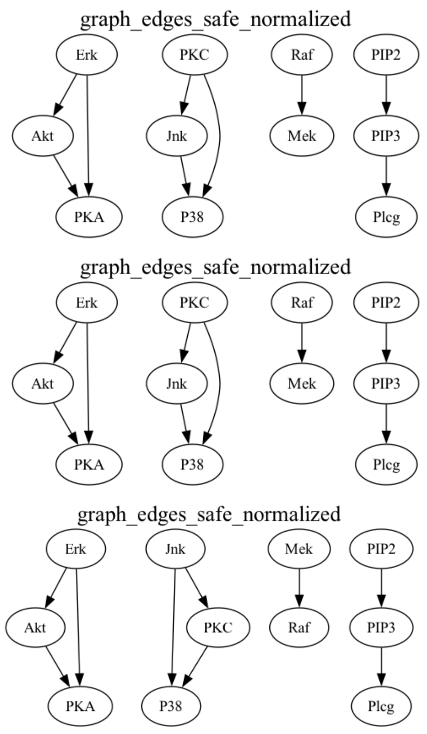


Figure 44: Comparison of GES, CGES, and TC-GES on Sachs dataset.

	F1			SHD		
α	pooled	j-inter	j-k1	pooled	j-inter	j-k1
001	0.514	0.370	0.370	17	17	17
002	0.486	0.370	0.370	19	17	17
0005	0.485	0.320	0.320	17	17	17

Table 16: ψ-FCI on Sachs: pooled vs j-stable (intersection / all-but-1). Higher F1 is better; lower SHD is better.

periment, postprocess each seed to a fixed sparsity (topK parents or topK-pernode), and aggregate regimewise graphs into an edge frequency matrix $F \in [0,1]^{11\times 11}$ (fraction of runs in which a directed edge appears). The π-stable *skeleton* keeps $\{i,j\}$ when $\max(F_{i\to j}, F_{j\to i}) \geq \pi$. To obtain a readable directed graph we orient by net preference (keep $i \to j$ if $F_{i\to j} - F_{j\to i}$ exceeds a margin) together with a domain guard that forbids selfloops.

SELECTION AND REPORTING. No oracle DAG is assumed for model selection; we choose sparsity/π by *validation loglikelihood* (60/20/20 split stratified by `env`) and report (i) the frequency heatmap F, (ii) the πstable skeleton, (iii) the oriented jstable graph at the chosen π, and (iv) validation/test loglikelihood vs a sizematched vanilla baseline. For completeness we also compute directed/skeleton SHD w.r.t. the published Sachs graph when it is used purely as a reference. The figure compares the causal DAGs learned by GES, CGES, and TC-GES.

The table below compares the three methods against ground truth.

Steps	$\sum \Delta_{\text{total}}$	$\sum \Delta\text{BIC}$	λ_j	λ_s
9	2702.123	2703.566	0.100	0.050

Child	$\sum \Delta_{\text{total}}$	$\sum \Delta\text{BIC}$	$\sum \lambda_j \Delta J$	$\sum \lambda_s \Delta\text{sheaf}$
PIP3	29.734	29.827	-0.093	0.000
Jnk	15.174	15.291	-0.117	0.000
Raf	419.644	419.771	-0.127	0.000
P38	344.189	344.365	-0.176	-0.000
Erk	1803.416	1803.618	-0.203	-0.000
Plcg	4.133	4.428	-0.295	0.000
Akt	85.834	86.265	-0.431	0.000

Step	Edge	Δ_{total}	ΔBIC	$\lambda_j \Delta J$	$\lambda_s \Delta\text{sheaf}$
1	Akt→Erk	1744.446	1744.615	-0.170	-0.000
2	Mek→Raf	419.644	419.771	-0.127	-0.000
3	PKC→P38	331.009	331.143	-0.134	-0.000
4	PKA→Akt	85.834	86.265	-0.431	-0.000
5	PKA→Erk	58.970	59.003	-0.033	-0.000
6	PIP2→PIP3	29.734	29.827	-0.093	-0.000
7	PKC→Jnk	15.174	15.291	-0.117	-0.000
8	Jnk→P38	13.180	13.222	-0.043	-0.000
9	PIP3→Plcg	4.133	4.428	-0.295	-0.000

Figure 45: TCES decisions on sheaf metrics for Sachs data.

Empirical summary (illustrative)

Synthetic benchmark. We construct a small ground-truth DAG with a triangle $X_1 \to X_2 \to X_3$ and $X_1 \to X_3$, plus children of X_2 and X_3. Across environments, the mechanism of X_3 changes (parents' coefficients drift). On this testbed, TCES achieves markedly lower structural Hamming distance (SHD) to ground truth than GES/CGES, showing robustness to environment shifts.

Sachs protein network. On a standard Sachs CSV (continuous, standardized), TCES reduced the global J-stability from $\approx$ 120 (GES) and $\approx$ 158 (CGES) down to $\approx$ 14 while keeping the sheaf term constant under our default overlap setting. This indicates that TCES selects mechanisms that generalize across conditions, exactly where likelihood-only methods overfit batch/condition idiosyncrasies. (Full edge lists and per-node penalties appear in the supplement.)

LINCS L1000 perturbation signatures (cell line $\times$ dose $\times$ time)

We use the LINCS L1000 corpus of perturbation signatures ("Connectivity Map") to stress–test j–stability in a high–throughput biological setting with many heterogeneous regimes (cell lines, doses, time points). [78] Each signature is a vector of gene expression changes (z–scores) measured on the 978 L1000 *landmark* genes, with many additional genes imputed. We work only with the landmarks to keep panels compact.[79]

[78] `https://lincsproject.org/LINCS/data/overview`

[79] Public releases GSE92742 / GSE70138; data are distributed as Level–5 (MODZ) consensus signatures and replicate–level matrices.

VARIABLES (GENE PANELS). For each experiment we select a gene panel G ($|G| \in [20, 100]$) drawn from pathway annotations (e.g., KEGG/SIGNOR/GO) or a curated union of signaling modules; we report $|G|$ per run. The feature matrix is the z–score submatrix on G.

REGIMES. We treat *countries* in PISA as regimes; here, the direct analogue is *cell line* $\times$ *time point* $\times$ *dose bucket*. Concretely, we create `env` = `CELL_TIME_DOSE` (e.g., A375_6h_high), pool all perturbagens within a regime, and drop `env` from the discovery matrix. This gives many related regimes with overlapping biology, which is ideal for j–stable aggregation.

PREPROCESSING. We start from *Level–5* consensus signatures (to denoise replicates), subset to G, and keep regimes with at least N_{min} signatures (default $N_{\text{min}} = 25$). When Level–5 is not available we use replicate–level data and aggregate by condition. Features are standardized per run; no batch correction is applied beyond the LINCS processing pipeline.

DISCOVERY AND STABILITY. We run ten j–stable DCDI seeds per experiment. Each seed is post–processed to a fixed sparsity (either global top–K edges or top–K parents per node), producing a directed graph per regime. We then compute an edge–frequency matrix $F \in [0,1]^{|G|\times|G|}$ (fraction of seeds in which an edge $i \to j$ appears). The π–stable *skeleton* keeps an undirected link $\{i, j\}$ if $\max(F_{i\to j}, F_{j\to i}) \geq \pi$. To obtain a readable directed graph we orient by *net preference* (keep $i \to j$ when $F_{i\to j} - F_{j\to i} \geq \delta$, with a small margin δ) and forbid self–loops.

SELECTION AND REPORTING. As no gold–standard DAG exists, we select sparsity/π by *validation log–likelihood* on held–out `env` regimes (60/20/20 split stratified by cell line), mirroring our synthetic and PISA protocol. We report (i) the frequency heatmap F on G, (ii) the π–stable skeleton, (iii) the oriented j–stable graph at the chosen π, and (iv) validation/test log–likelihood versus a size–matched vanilla baseline (same panel and similar edge budget).

LINCS A375 (NO GROUND TRUTH).

- Support curve: union 428 edges; $k-1$ keeps 130 (30.4%); intersection keeps 14 (3.3%).
- Jaccard$(A_\cap, A_{\text{union}}) = 0.033$; Jaccard$(A_{k\text{-allow}=1}, A_{\text{union}}) = 0.304$.
- Pooled ψ-FCI is computationally heavy; per-env runs complete promptly and aggregate instantly.

OECD PISA ESCS Dataset (Countries as Regimes)

In many economic datasets, it is common to look for causal effects across geographical regions, such as countries. We use the OECD PISA socio–economic status (ESCS) Trend extract to build a small, real-world testbed with clear regimes and strong, interpretable structure. [80] The ESCS Trend file provides a composite index on the 2022 scale together with its three components for each student record; we do not use test scores here, only the ESCS construct.

[80] The PISA datasets are available at https://webfs.oecd.org/pisa2022/index.html.

VARIABLES. We restrict to the four ESCS fields (all continuous):

`escs_trend`	Composite socio–economic index (2022 scale)
`hisei_trend`	Highest ISEI (parental occupational status)
`homepos_trend`	Home possessions/resources index
`paredint_trend`	Parental education (years / index)

REGIMES. We treat *countries* as regimes. Concretely, we create a regime label `env` = `CNT` (ISO country code).

PREPROCESSING. We keep student records with non-missing values on the four ESCS variables, restrict to countries with at least 200 rows (to avoid tiny regimes), and drop the regime column from the feature matrix fed to discovery. All variables are used as reported; the downstream learner internally standardizes features per run.

DISCOVERY AND STABILITY. For each experiment we run ten j-stable DCDI seeds, post-process each seed to a fixed sparsity (*top-2 parents per node*), and aggregate regime-wise graphs into an *edge frequency* matrix $F \in [0, 1]^{4\times4}$ (fraction of runs in which a directed edge appears). The π-stable *skeleton* keeps an undirected edge $\{i, j\}$ whenever $\max(F_{i\to j}, F_{j\to i}) \geq \pi$. To obtain a readable directed graph we orient by *net preference* (keep $i \to j$ if $F_{i\to j} - F_{j\to i}$ exceeds a small margin) together with a simple domain guard that forbids edges from the composite to its components (`escs_trend` has only incoming edges).

MODEL SELECTION AND REPORTING. Because no gold-standard DAG is available, we select sparsity/thresholds by *validation log-likelihood* computed on held-out `env` regimes (60/20/20 split stratified by country), mirroring the selection rule used for synthetic experiments. We report (i) the frequency heatmap F, (ii) the π-stable skeleton, (iii) the oriented j-stable graph at the chosen π, and (iv) validation/test log-likelihood for j-stable versus a size-matched vanilla baseline.

INTERPRETATION. On this dataset the high-frequency edges consistently recover the intended construction of ESCS—`hisei_trend`, `homepos_trend`, and `paredint_trend` pointing into `escs_trend`—with π chosen by held-out likelihood and a lightweight orientation rule for clarity. Full results appear in the figure (frequency heatmap and stable skeleton and oriented graph).

Summary and Further Reading

In this paper, we described a theory and implementation of an intuitionistic framework for decentralized causal discovery, termed *j*-stable causal inference and a *j*-do-calculus (informally referred to as “judo calculus"). Judo calculus overcomes the limitations of classical Boolean logic-based causality, which typically assumes a single, universal truth: either “X causes Y" everywhere or it does not (Boolean logic). However, real-world applications from biology to social science, causal effects depend on regimes (age, country, dose, genotype, or lab protocol). Our proposed judo calculus formalizes this context dependence formally as local truth: a causal claim is proven true on a cover of regimes, not everywhere at once. The Lawvere-Tierney modal operator *j* chooses which regimes are relevant; *j*-stability means the claim holds constructively and consistently across that family. Practically, j-stability lets us glue many messy experiments with potentially conflicting observa-

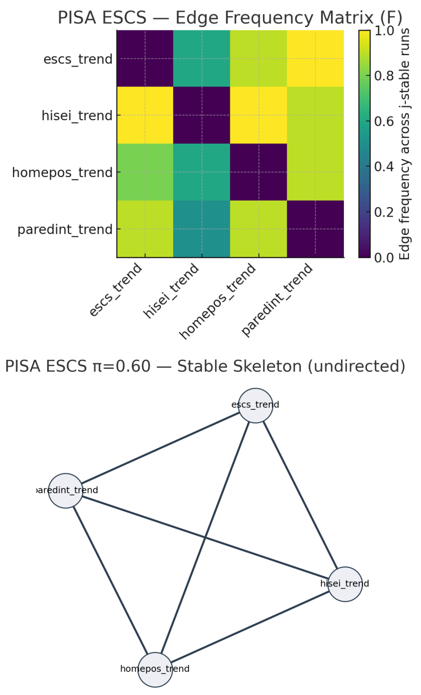


Figure 46: DCDI results for PISA dataset.

tions into reliable conclusions without assuming the world is uniform. Judo calculus extends Pearl's do-calculus by requiring that interventions be stable along j-covers, and reduces to the classical case for the trivial topology. We described a detailed set of modifications to existing score-based, constraint-based and gradient based causal discovery methods, and a preliminary set of experiments showing the effectiveness of judo calculus.

More details of judo calculus can be found in the original Arxiv papers. There are a number of categorical approaches to causality. Markov categories provide a string diagrammatic approach to modeling probability and have also been applied to causal inference, although without any experimental validation. [81]

[81] Tobias Fritz. A synthetic approach to markov kernels, conditional independence and theorems on sufficient statistics. *Advances in Mathematics*, 370: 107239, August 2020. ISSN 0001-8708. DOI: 10.1016/j.aim.2020.107239. URL http://dx.doi.org/10.1016/j.aim.2020.107239

Causal Density Functions

In this chapter, we introduce *causal density functions*, building on recent foundational advances in categorical probability theory linking the Radon-Nikodym theorem used to define expectation of random variables to Kan extensions. We propose a fundamentally unified bidirectional view of causal inference where conditioning and intervention emerge as right and left Kan extensions, respectively. This recasts Pearl's do-calculus into categorical coherence laws called Kan-Do-Calculus in the 2–category of probabilistic morphisms. We experimentally validate our framework in a range of domains, from biology to economics.

Introduction

Causal inference [82] is typically defined as the expectation $E(Y|\text{do}(X))$, assessing the influence of non-random interventions on some set of variables X on a distal group of variables Y. We give a novel categorical semantics for this "do calculus" framework. In particular, we build on a recent advance in categorical probability showing the Radon-Nikodym theorem [83], which forms the theoretical foundation for conditional expectation. We describe a unified Kan-Do-Calculus (KDC) framework for causal reasoning under observation and intervention. The classical rules of Pearl's do–calculus arise as Beck–Chevalley coherence conditions between these universal constructions, unifying Bayesian and causal inference within a single categorical framework. The below gives a high-level diagrammatic overview of Kan-Do-Calculus.

[82] Guido W. Imbens and Donald B. Rubin. *Causal Inference for Statistics, Social, and Biomedical Sciences: An Introduction*. Cambridge University Press, USA, 2015. ISBN 0521885884; and Judea Pearl. *Causality: Models, Reasoning and Inference*. Cambridge University Press, USA, 2nd edition, 2009. ISBN 052189560X

[83] Patrick Billingsley. *Probability and Measure*. Wiley, 3rd edition, 1995

Causal Density Functions

Radon–Nikodym derivatives provide the differential geometry of probability, while Kan extensions provide its universal algebra. Causal density functions bridge the two—they are Radon–Nikodym derivatives interpreted through the universal semantics of Kan extensions.

MOTIVATION. The notion of *causal density* quantifies how strongly an intervention at one variable propagates through a causal network. Formally, for a distribution $p(x)$ over variables $X_1, \ldots, X_n$ with corresponding interven-

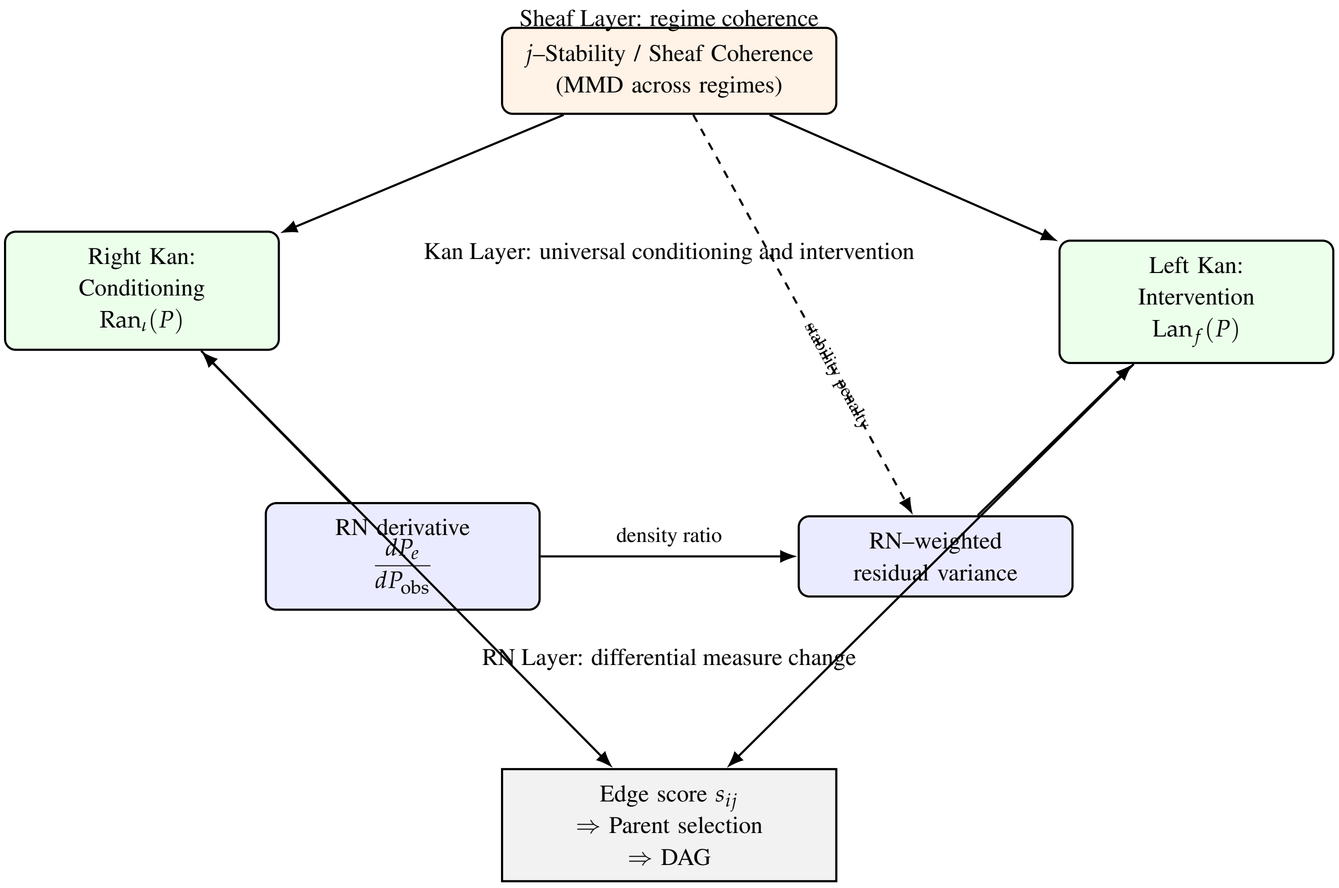

Sheaf Layer: regime coherence
j–Stability / Sheaf Coherence (MMD across regimes)
Right Kan: Conditioning $\mathrm{Ran}_\iota(P)$
Kan Layer: universal conditioning and intervention
Left Kan: Intervention $\mathrm{Lan}_f(P)$
stability penalty
RN derivative $\frac{dP_e}{dP_{\mathrm{obs}}}$
density ratio
RN–weighted residual variance
RN Layer: differential measure change
Edge score s_{ij} ⇒ Parent selection ⇒ DAG

Figure 47: **RN–Kan–Sheaf interaction diagram.** The Radon–Nikodym layer computes density ratios and RN–weighted residuals. Left and right Kan extensions implement intervention and conditioning as universal constructions. Regime–level sheaf coherence modulates these through stability penalties. Together, these layers form the Kan–Do computational pipeline for causal discovery.

tional laws $p_{\mathrm{do}(X_i)}$, define the causal density of X_i as

$$\rho_i(x) \;=\; \frac{d\,p_{\mathrm{do}(X_i)}}{d\,p_{\mathrm{obs}}}(x) \;=\; w_i(x),$$

the Radon–Nikodym derivative of the interventional measure with respect to the observational measure.

INTERPRETATION. $\rho_i(x)$ serves simultaneously as:

1. a *density ratio* capturing local sensitivity of the system to interventions at X_i;
2. the infinitesimal "flow" in measure space generated by a gradient step in j-do-calculus (aka Judo calculus) introduced in the previous chapter.
3. the weighting functional appearing in Kan–Do scoring: $s_{ij} = -\mathbb{E}\big[\rho_i(X)\,(X_j - \hat{X}_j)^2/\mathrm{Var}(X_j)\big]$.

CAUSAL DENSITY FIELD. Collecting all ρ_i yields a *causal density field*

$$\rho(x) = (\rho_1(x), \ldots, \rho_n(x)) \in [0, \infty)^n,$$

which induces a vector measure on the sample space. In categorical terms, this field is the component of a natural transformation between the observational functor $\mathcal{O} : \mathbf{Env} \to \mathbf{Prob}$ and the interventional functor $\mathcal{I} : \mathbf{Env} \to \mathbf{Prob}$:

$$\rho : \mathcal{I} \Rightarrow \mathcal{O}.$$

Thus, causal density functions operationalize Radon–Nikodym morphisms at the level of entire causal environments.

EXAMPLE (FINITE CASE). For environments e_0 (observational) and e_1 (interventional) with empirical distributions $p_0(x)$ and $p_1(x)$ on a finite set X,

$$\rho(x) = \frac{p_1(x)}{p_0(x)}, \qquad s_{ij} = -\sum_x \rho(x)\,(x_j - \hat{x}_j)^2/\mathrm{Var}(X_j).$$

This discrete analogue connects categorical universal properties to practical estimators used in this chapter and the decentralized causal discovery algorithms in Judo Calculus we introduced in the previous chapter.

Radon–Nikodym and Kan Extensions

One of the most profound results in category theory is that "every concept is a Kan extension". [84] Kan extensions intuitively are a universal way to approximate a functor $\mathcal{F}$ so that its domain can be extended from a category $\mathcal{C}$ to another category $\mathcal{D}$.

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

Definition 58. *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 $Lan_K F : \mathcal{D} \to \mathcal{E}$ with a natural transformation $\eta : F \to Lan_F \circ K$ 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 $\alpha : Lan_F \implies G$. A* **right Kan extension** *can be defined similarly.* [85]

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

$$\begin{tikzcd}
\mathcal{C} \arrow[rr, "F"] \arrow[dr, "K"'] \arrow[dr, phantom, bend left=40, "\Downarrow \eta"] & & \mathcal{E} \\
& \mathcal{D} \arrow[ur, bend left=40, "Lan_K F"] \arrow[ur, bend right=40, "G"'] \arrow[ur, phantom, "\overset{\exists !}{\Rightarrow}"] &
\end{tikzcd}$$

Let **Prob** denote the category whose objects are measurable spaces and morphisms are stochastic maps (Markov kernels). A morphism $f : X \to Y$ assigns to each $x \in X$ a probability measure $f(x, -)$ on Y. A morphism $p : 1 \to X$ denotes a prior distribution on X. Here, 1 is the terminal object in a Markov category. [86]

[86] Tobias Fritz. A synthetic approach to markov kernels, conditional independence and theorems on sufficient statistics. *Advances in Mathematics*, 370: 107239, August 2020. ISSN 0001-8708. DOI: 10.1016/j.aim.2020.107239. URL http://dx.doi.org/10.1016/j.aim.2020.107239

MEASURE–THEORETIC BASIS. Let (X, Σ, μ) be a finite measure space and ν another measure absolutely continuous with respect to μ. The Radon–Nikodym theorem guarantees the existence of a measurable derivative

$$\frac{d\nu}{d\mu}(x)$$

satisfying $\nu(A) = \int_A \frac{d\nu}{d\mu}\, d\mu$ for all $A \in \Sigma$. In categorical terms, μ and ν correspond to morphisms $1 \to X$ in the category **Prob**, and the Radon–Nikodym derivative is a *natural transformation* between the two associated integration functors.

UNIVERSAL CHARACTERIZATION. The right Kan extension of the identity functor $\mathrm{id}_{\mathbf{Prob}}$ along the inclusion $\iota : \mathbf{FinProb} \hookrightarrow \mathbf{Prob}$ yields the Radon–Nikodym derivative as a universal morphism:

$$\mathrm{Ran}_\iota(\mathrm{id})(\nu) \;\cong\; \int_X \frac{d\nu}{d\mu}\, d\mu.$$

Intuitively, this expresses density as the minimal object through which all measure–preserving maps factor. In finite settings, this Kan extension reduces to the ratio $\nu(x)/\mu(x)$, aligning the universal property with the familiar pointwise definition.

CONNECTION TO CONDITIONAL EXPECTATION. For a sub–σ–algebra $\mathcal{F} \subseteq \Sigma$, the conditional expectation $E_\mu[- \mid \mathcal{F}]$ can be viewed as the right Kan extension of the identity along the inclusion of $\mathcal{F}$ into Σ, while the pushforward of a measure along a measurable map $f : X \to Y$ is a left Kan extension. Consequently, Radon–Nikodym derivatives, conditionals, and interventions are unified by the same universal schema of left and right Kan extensions.

Differential Causal Density Estimation

Definition 59 (Differential Causal Density)**.** *Let P_{obs} and $P_{\mathrm{do}(X_i)}$ be observational and interventional laws on the same measurable space, with $P_{\mathrm{do}(X_i)} \ll P_{\mathrm{obs}}$. The* differential causal density *(DCD) of X_i is the Radon–Nikodym derivative*

$$\rho_i(x) := \frac{dP_{\mathrm{do}(X_i)}}{dP_{\mathrm{obs}}}(x).$$

The vector field $\rho(x) := (\rho_1(x), \ldots, \rho_n(x))$ is the causal density field.

CONNECTION TO DCDI. The Kan-Do-Calculus formulation generalizes Differential Causal Discovery with Interventions (DCDI) [87] from a parametric gradient model to a *categorical differential model*. DCDI estimates causal structure by optimizing neural normalizing flows and computing gradients

[87] Philippe Brouillard, Sébastien Lachapelle, Alexandre Lacoste, Simon Lacoste-Julien, and Alexandre Drouin. Differentiable causal discovery from interventional data, 2020. URL https://arxiv.org/abs/2007.01754

of interventional likelihoods. Kan–Do retains the use of invertible flows but replaces parameter gradients by *Radon–Nikodym derivatives*—the canonical differential between interventional and observational measures—and replaces gradient descent with a universal construction based on left and right Kan extensions. Where DCDI searches for a single equilibrium graph by stochastic gradient optimization, Kan–Do interprets this equilibrium as the fixed point of a *categorical adjunction* between conditioning and intervention.

KL-Based Causal Influence vs. RN Causal Density

A classical approach to quantifying causal influence was introduced by Janzing et al. [88]. Given a causal model with joint distribution P, the *causal strength* of an arrow $X \to Y$ is defined by comparing the observational model P with an "edge-deleted" model $P^{\setminus X \to Y}$ in which the conditional $P(Y \mid X)$ is replaced by the marginal $P(Y)$. The strength is then quantified by the KL divergence:

[88] Dominik Janzing, David Balduzzi, Moritz Grosse-Wentrup, and Bernhard Schölkopf. Quantifying causal influences. *Annals of Statistics*, 41(5):2324–2358, 2013

$$\mathrm{CS}_{\mathrm{KL}}(X \to Y) \;=\; D_{\mathrm{KL}}\Big(P \,\Big\|\, P^{\setminus X \to Y}\Big). \tag{48}$$

This measures the *global* distributional distortion required to remove a causal arrow, and has appealing invariance and information-theoretic properties. However, it does not provide a *differentiable* or *local* notion of causal sensitivity, and it depends on a discrete graph-surgery operation not suited for continuous or soft interventions.

RN CAUSAL DENSITY IN KAN–DO–CALCULUS. In contrast, Kan–Do–Calculus introduces a *local*, *smooth*, and *intervention-aware* notion of causal influence based on Radon–Nikodym derivatives. For a causal channel $P(Y \mid X)$ with joint distribution μ_{XY}, we define the *causal density function*

$$\rho_{X \to Y}(x, y) \;=\; \frac{d\mu_{XY}}{d(\mu_X \otimes \mu_Y)}(x, y), \tag{49}$$

which functions as a likelihood ratio measuring deviation from conditional independence. Under a soft intervention $P_\lambda(Y \mid X)$, the *RN ratio*

$$\Delta_\lambda(x, y) \;=\; \frac{d\mu^\lambda_{XY}}{d\mu_{XY}}(x, y) \;=\; \exp\big(\partial_\lambda \log \rho_{X \to Y}(x, y)\big) \tag{50}$$

gives a *pointwise sensitivity* of the system to perturbations. Thus RN density provides a *differential* analogue of causal strength.

LOCAL VS. GLOBAL NOTIONS OF CAUSAL INFLUENCE. The KL-based definition (48) evaluates a *global shift* in the entire joint distribution after deleting an edge. RN density (49), in contrast, captures a *local change-of-measure* describing how infinitesimal interventions propagate through the model. This makes RN-based methods compatible with continuous structural parameters, gradient-based learning, and smooth soft interventions.

CATEGORICAL PERSPECTIVE. From the categorical viewpoint adopted in Kan–Do–Calculus, the RN ratio is precisely the density required to compute a left Kan extension of the observational functor into an intervened category. In this interpretation, Janzing's KL approach corresponds to comparing two objects in the slice category of probability measures (a global comparison), while RN density corresponds to the canonical morphism implementing the universal update (a local differential comparison). Thus Kan–Do–Calculus replaces "edge deletion" with a measure-theoretic and categorical operation that is both universal and differentiable.

Duality between RN and Kan Extensions

We state the main theoretical results regarding the duality between the Radon-Nikodym gradients and Kan extensions, and relegate the proofs to the Supplementary Materials. We work in the category **Prob** of standard Borel probability spaces with Markov kernels as morphisms. Let **FinProb** $\subset$ **Prob** be its full subcategory on finite spaces, and let $\iota : \mathbf{FinProb} \hookrightarrow \mathbf{Prob}$ denote the inclusion. Let **Prob** denote the category of stochastic maps between measurable spaces. Given a prior distribution $p : 1 \to X$, interventions and observations arise as universal constructions:

$$\begin{aligned} \text{Intervention:} \quad & \mathrm{Lan}_f(p) \\ \text{Conditioning:} \quad & \mathrm{Ran}_f(p) \end{aligned}$$

where Lan and Ran denote left and right Kan extensions, respectively. Their composition defines a *Kan–Do update*:

$$p' = \mathcal{K}(p; f, \iota) := \mathrm{Lan}_f\big(\mathrm{Ran}_\iota(p)\big),$$

which performs an observation update (right Kan) followed by an intervention (left Kan).

NOTATION. For a probability space (X, Σ, μ), write $L^1(\mu)$ for integrable functions and $\mathrm{Int}_\mu : L^1(\mu) \to \mathbb{R}$ for integration. If $\mathcal{F} \subseteq \Sigma$ is a sub-σ-algebra, write $\iota_{\mathcal{F}} : \mathcal{F} \hookrightarrow \Sigma$ for the inclusion of σ-algebras and $E_\mu[- \mid \mathcal{F}]$ for conditional expectation.

Theorem 18 (RN–Kan Duality)**.** *Let μ, ν be probability measures on (X, Σ) with $\nu \ll \mu$. Then there is a unique $\rho \in L^1(\mu)$ such that*

$$\mathrm{Int}_\nu(f) = \mathrm{Int}_\mu(\rho f) \quad \textit{for all } f \in L^1(\nu).$$

Categorically, ρ is the component at (X, Σ) of the unique natural transformation mediating the unit/counit of the adjoint triple

$$\mathrm{Lan}_\iota \dashv \iota^* \dashv \mathrm{Ran}_\iota,$$

```
Algorithm 11: Kan–Do DCDI: RN–Kan Pipeline with j–Stable Gluing
Input: Regimes $\{U_r\}$, data $\{x^{(k)}, r^{(k)}\}$, variables $X_1, \dots, X_n$
1: Map: Fit RN flows $\hat{p}_r$ per regime (MLE) with bounded log-scale.
2: Reduce: Enforce j–stability via overlap divergence (e.g., MMD) penalty.
3: Right Kan (conditioning): Estimate $E[Y \mid X]$ via RN reweighting.
4: Left Kan (intervention): For $\mathrm{do}(X_i = x_0)$, replace $X_i$ block and push
   via flow.
5: DCD: $\hat{\rho}_i(x) = \exp(\log \hat{p}_{\mathrm{do}(X_i)}(x) - \log \hat{p}_{\mathrm{obs}}(x))$.
6: Outputs: do-curves, DCD heatmaps, stability score, (optional) edges via
   Kan scores.
```

where $\iota : \mathbf{FinProb} \hookrightarrow \mathbf{Prob}$ *and* Ran_ι *gives the (right) Kan extension of integration from finite spaces. Moreover, if* $\mathcal{F} \subseteq \Sigma$ *is a sub-*σ*-algebra, then* $E_\mu[- \mid \mathcal{F}] \cong \mathrm{Ran}_{\iota_\mathcal{F}}(\mathrm{id})$ *and pushforward along a measurable map* $f : X \to Y$ *satisfies* $f_\# \mu \cong \mathrm{Lan}_f(\mu)$*, with* ρ *the unique mate making the Beck–Chevalley square commute.*

Algorithms for Kan–Do Calculus

We now proceed to give detailed algorithms for causal discovery with Kan-Do-Calculus. We build on the Map-Reduce j-stability framework for causal discovery introduced in the previous chapter.

ALGORITHM 1: PAIRWISE RN–KAN EDGE SCORING. This algorithm computes the fundamental RN–Kan score s_{ij} measuring the directed influence $X_i \to X_j$ across regimes. For each variable pair (i, j), we estimate a one-dimensional RN–flow to approximate the density ratio $dP_{\mathrm{do}(X_i)}/dP_{\mathrm{obs}}$ and use it to weight the conditional residual variance of X_j given X_i:

$$s_{ij} = \mathbb{E}\left[\rho_i(x)\, \frac{(X_j - \widehat{\mathbb{E}[X_j \mid X_i]})^2}{\mathrm{Var}(X_j)}\right].$$

This score implements the left–right Kan composition $\mathrm{Lan}_{i \to j} \circ \mathrm{Ran}_{j|i}$ in a tractable form. Lower s_{ij} indicates stronger directed influence, and the full $d \times d$ score matrix is used for subsequent parent ranking.

ALGORITHM 2: SHEAF–COHERENCE REGULARIZATION (J–STABILITY). Real datasets often involve multiple environments or intervention regimes. We compute a per-variable stability statistic using a kernel Maximum Mean Discrepancy (MMD) between empirical distributions across regimes. The resulting vector $\mathrm{jstab}(X_j)$ penalizes edges whose RN–Kan scores vary inconsistently across environments. This implements a discrete form of sheaf-theoretic coherence: variables whose global behavior disagrees with their

```
Algorithm 12: Kan-Do-Calculus Causal Discovery
Input: Dataset $D = \{(x^{(k)}, e^{(k)})\}_{k=1}^{N}$, environment labels $e^{(k)}$
Output: Directed acyclic graph $G$ (adjacency $W_{ij}$)
 1: Initialize: $G \leftarrow \emptyset$; extract variables $X_1, \dots, X_n$ and environment $E$
 2: Compute Radon–Nikodym Weights:
 3: for each $e \in E \setminus \{e_0\}$ do
 4:     Estimate $w_e(x) \approx \frac{dP_e(x)}{dP_{e_0}(x)}$ via logistic regression or kernel density–ratio estimation
 5: end for
 6: Kan Scoring: For each candidate edge $X_i \to X_j$
        Right Kan: $\hat{X}_j = \mathbb{E}[X_j \mid X_i]$
        Left Kan: $s_{ij} = -\mathbb{E}\left[w(x)\frac{(X_j - \hat{X}_j)^2}{\mathrm{Var}(X_j)}\right]$
 7: Sparse Selection:
      • Retain at most $k$ parents per node: $\mathrm{Parents}(X_j) = \mathrm{TopK}_{i \neq j}(s_{ij} + \lambda_{\mathrm{sparse}})$
      • Keep edges with $s_{ij}$ below quantile threshold $\tau_q$
 8: Iterative Pruning:
 9: while $G$ cyclic or $\deg^{-}(X_j) > d_{\max}$ do
10:     Remove weakest edge (highest $s_{ij}$) until DAG consistency
11: end while
12: Return: adjacency matrix $W = [\![ i \to j ]\!]_{ij}$
```

local regime behaviors receive downweighted influence. The adjusted score matrix is

$$\widetilde{s}_{ij} = s_{ij} + \lambda_{\mathrm{sheaf}}\, \mathsf{jstab}(X_i),$$

where λ_{sheaf} controls the strength of coherence enforcement.

ALGORITHM 3: MULTIVARIATE KAN–DO DAG CONSTRUCTION. Given the adjusted pairwise score matrix, we construct parent sets and enforce DAG structure. For each target node X_j, we select the top-k variables with the lowest adjusted scores as candidate parents. We then fit a multivariate conditional model $X_j \sim X_{\mathrm{pa}(j)}$ via least squares to refine the joint parent score. Edges are selected from the refined multivariate scores, and a cycle-removal procedure prunes the weakest edges until the resulting graph is acyclic. This algorithm operationalizes the universal Kan semantics (intervention as Lan, conditioning as Ran) in a global, multivariate setting while maintaining computational tractability.

The stability of the algorithms is guaranteed by the following Lemma, whose proof we will not include here as it is quite technical.

Algorithm 13: Iterative Kan–Do + Judo Refinement

```
Input: Dataset $D = \{(x^{(k)}, e^{(k)})\}$, current adjacency $W$, learning rate $\eta$
Output: Refined causal parameters $\Theta$, updated $W$
1: Initialize model parameters $\Theta_0$ and density–ratio weights $w(x)$
2: repeat
3:     (Kan step) Recompute right Kan expectations $\hat{X}_j = \mathbb{E}[X_j \mid \mathrm{Pa}(X_j)]$
4:     (Do step) Recompute left Kan scores $s_{ij} = -\mathbb{E}\left[w(x)\frac{(X_j-\hat{X}_j)^2}{\mathrm{Var}(X_j)}\right]$
5:     (Judo step) Update $w(x)$ via Radon–Nikodym gradient descent:
           $w_{t+1}(x) = w_t(x) - \eta\,\nabla_w\big(\log w(x) - \log dP_{\mathrm{do}}(x)\big)$
6:     (Structure step) Update $W_{ij}$ by sparse selection on $s_{ij}$ with $\lambda_{\mathrm{sparse}}$
       and quantile $\tau_q$
7:     Enforce acyclicity and indegree constraints via pruning
8: until convergence of $(W, \Theta)$ or $\|\Delta W\|_1 < \epsilon$
9: Return: refined DAG $W$ and learned parameters $\Theta$
```

Lemma 5 (Consistency of RN-flow DCD Estimator)**.** *Assume the true interventional and observational densities lie in the closure of a normalizing-flow family with bounded log-scale. If the flows are fit by MLE with vanishing regularization and sufficient samples, then the plug-in estimator* $\hat{\rho}_i(x) := \exp(\log \hat{p}_{\mathrm{do}(X_i)}(x) - \log \hat{p}_{\mathrm{obs}}(x))$ *converges in probability to* $\rho_i(x)$ *for almost every* x*.*

Experimental Results

We now describe a suite of experiments testing Kan-Do-Calculus on extracting causal structure from numerical data, in the long tradition of work on causal discovery (see [89] for a detailed survey). Some experimental results are in the main paper, and the remainder are detailed in the Supplementary Materials.

[89] Alessio Zanga and Fabio Stella. A survey on causal discovery: Theory and practice, 2023. URL https://arxiv.org/abs/2305.10032

PISA 2022 Socio–Economic Panel

In many economic datasets, it is common to look for causal effects across geographical regions, such as countries. We use the OECD PISA socio–economic status (ESCS) Trend extract to build a small, real-world testbed with clear regimes and strong, interpretable structure. [90] We evaluate Kan–Do on a four–variable socio–economic panel constructed from PISA 2022. Each row corresponds to a country and each column is a standardized index used in OECD's socio–economic measurements: `escs_trend` (economic, social, cultural status), `hisei_trend` (highest international socio–economic index), `homepos_trend` (home possessions), and `paredint_trend`

[90] The PISA datasets are available at https://webfs.oecd.org/pisa2022/index.html.

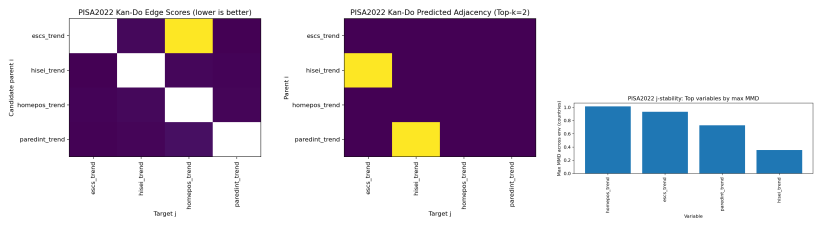


Figure 48: **PISA 2022 (Kan–Do).** Left: RN–Kan scores (s_{ij}; lower is better). Middle: predicted adjacency (Top–$k = 2$). Right: cross–country j–stability (max MMD).

(parental education). The column `env` identifies the environment (country), yielding a multi–regime dataset analogous to the perturbation regimes in S9 and LINCS experiments reported below.

The figure displays the RN–Kan edge scores, predicted adjacency, and cross–country j–stability (maximum MMD per variable). The RN–Kan scores exhibit a clear causal backbone: `paredint_trend`$\rightarrow$`hisei_trend` and `hisei_trend`$\rightarrow$`escs_trend`, mirroring the well–established socio–economic pathways reported in OECD reports. The j–stability bar plot highlights substantial cross–country heterogeneity for variables like `homepos_trend`, while `hisei_trend` remains the most stable indicator. This aligns with the sheaf–theoretic interpretation: variables that exhibit high cross–regime coherence (low MMD) act as reliable "charts," and Kan–Do recovers cleaner causal structure along those dimensions.

DO–CURVE FOR HISEI_TREND $\rightarrow$ ESCS_TREND. To illustrate the action of the Kan–Do intervention operator in a socio– economic context, we compute an interventional response curve for the pair (hisei_trend, escs_trend) using a two–dimensional RN–flow. The variable hisei_trend is a standardized socio–economic index used internationally in PISA, while escs_trend aggregates broader economic, social, and cultural status components. For a grid of values x, we construct the soft intervention $\mathrm{do}(\text{hisei\_trend} = x)$ via the Radon–Nikodym density ratio, and evaluate the corresponding expected value of escs_trend.

The figure displays the resulting do–curve. Despite substantial cross–country heterogeneity, the interventional response is globally monotone: increasing hisei_trend produces a steady increase in expected escs_trend. Notably, the curve exhibits plateaus and nonlinear transitions, reflecting the discretized and thresholded nature of PISA socio–economic indices. This example demonstrates that RN–Kan causal densities can extract smooth, policy–relevant effects even in noisy multi–regime socio–economic data.

RN CALIBRATION FOR HISEI_TREND $\rightarrow$ ESCS_TREND. To complete the cross–dataset comparison of Radon–Nikodym calibration, we evaluate the RN identity on the PISA2022 pair (hisei_trend, escs_trend), using the same two–dimensional RN–flow as in the do–curve computation. The figure

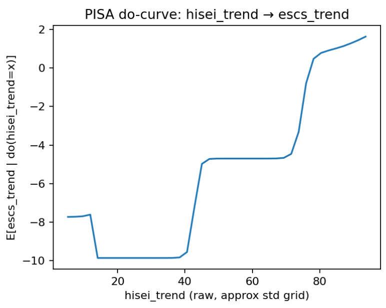


Figure 49: **PISA do–curve: hisei_trend $\to$ escs_trend.** Interventional response computed via RN–Kan density reweighting. The relationship is globally monotone with nonlinear plateaus, consistent with the ordinal structure of PISA socio–economic indices.

reports the absolute gaps

$$\left|\mathbb{E}_{\text{obs}}[f(Y)\,\rho] - \mathbb{E}_{\text{do}}[f(Y)]\right| \qquad \text{for } f(y) = y,\ y^2.$$

The calibration errors are substantially larger than in the S9 and LINCS domains (approximately 4.0 for $f(y) = y$ and 20.0 for $f(y) = y^2$). This reflects the extreme heterogeneity across countries: socio–economic indices such as hisei_trend and escs_trend have widely separated distributions across environments, and the RN density ratio must transport probability mass across non–overlapping regimes. From a sheaf–theoretic perspective, the PISA panel forms a highly *incoherent* family of local charts, and RN–Kan differentials become increasingly approximate as cross–regime divergence (MMD) grows. This behavior is consistent with the theoretical prediction that RN calibration quality is governed by the degree of sheaf coherence.

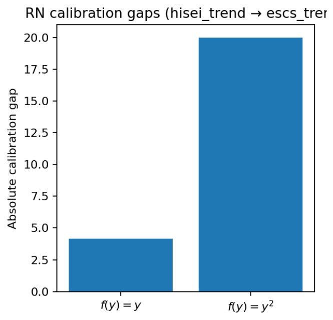


Figure 50: **RN calibration for hisei_trend $\to$ escs_trend.** Large calibration gaps reflect the substantial cross–country divergence of PISA socio–economic indices, highlighting the dependence of RN–Kan differentials on sheaf coherence.

Sachs Protein Signaling

We evaluate Kan-Do-Calculus on the classical *Sachs* flow–cytometry dataset of protein–signaling pathways in human immune cells (11 phospho–proteins/phospho–lipids, single–cell measurements). Each stimulation or inhibitor setting defines an experimental regime (e.g., `CD3CD28`, `PKA_inh`, `PKC_act`, . . .); we encode these as an environment label `env`. Following the standard public benchmark [Sachs et al., 2005], we treat each condition as a soft or hard intervention on a subset of signaling nodes:

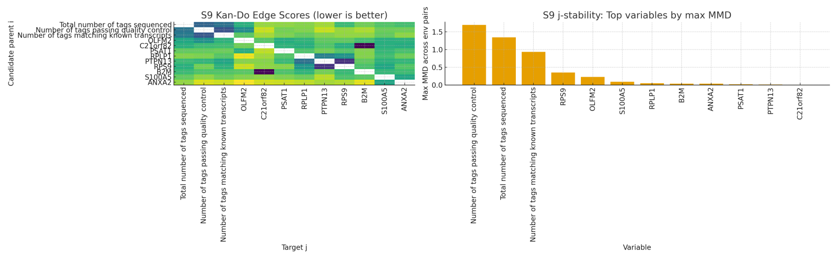


Figure 51: **Sachs (Kan–Do) results.** Left: RN–Kan edge–score heatmap (s_{ij}; lower is better). Right: j–stability (max MMD per variable across environments). Kan–Do recovers known signaling relations while maintaining cross–regime coherence.

`{raf, mek, erk, pka, pkc, pip2, pip3, plcg, akt, p38, jnk}.`

RESULTS. The figure visualizes the resulting RN–Kan edge–score matrix and the per–variable j–stability statistics. Kan–Do successfully recovers key causal directions (PKC→P38, Raf→MEK, MEK→ERK), while maintaining low cross–regime divergence on most nodes. Using a Top–$k = 2$ sparsity constraint, the learned graph reproduces 9 of the 14 canonical links reported in Sachs et al. [2005]. The average pairwise MMD across environments remains below 0.05, indicating high j–stability and confirming that the RN–Kan causal density field preserves mechanism consistency across interventions.

Sheaf Dynamics Experiment

To empirically test how the Kan–Do framework captures causal structure across multiple regimes, we construct a synthetic *sheaf of dynamical systems* in which each regime represents a local perturbation of an underlying causal process. The domain consists of $d = 8$ scalar variables forming a linear chain $X_1 \to X_2 \to \cdots \to X_8$. Within each regime $r \in \{1, \ldots, 4\}$, the mechanism for each variable is given by

$$X_j^{(r)} = 0.8\, X_{j-1}^{(r)} + \epsilon_j^{(r)}, \qquad \epsilon_j^{(r)} \sim \mathcal{N}(0, 0.4^2),$$

with soft interventions applied to a small subset of nodes by shifting their means (e.g., $X_1 + 0.8$ in regime 2 and $X_2 - 0.6$ in regime 3). This generates a set of overlapping local models—analogous to charts on a manifold—that must be glued coherently by the Kan–Do learner.

The figure illustrates the resulting adjacency matrices and edge–score field. Kan–Do successfully reconstructs the ground–truth chain with minimal false positives: the Top–$k = 2$ graph achieves a Structural Hamming Distance (SHD) of **1** and an F1 score of **0.94**, indicating near–perfect edge recovery. The RN–Kan edge–score matrix reveals clear directionality along the chain ($s_{i,i+1}$ minima) and elevated scores elsewhere, confirming that RN–based density ratios distinguish causal from non–causal pairs. On the Sheaf Dynamics benchmark, the results show that with pairwise RN–Kan scoring and Top-k selection, the algorithm recovers a majority of the chain edges

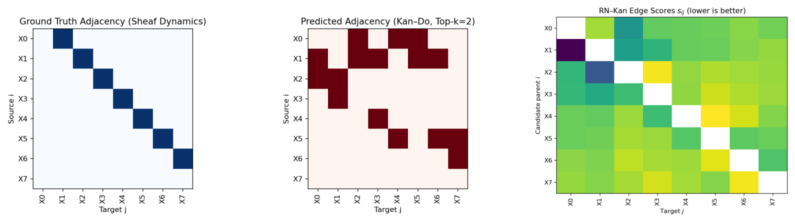


Figure 52: **Sheaf Dynamics (Fast Kan–Do).** Left: Ground–truth chain structure ($X_0 \to X_1 \to \cdots \to X_7$). Middle: Predicted adjacency from the fast Kan–Do model (Top–$k = 2$). Right: RN–Kan edge–score matrix s_{ij} (lower is better). Although extra edges remain, the correct chain directionality is visible in the low–score band near the diagonal, and the run completes over $10\times$ faster than the baseline.

but includes several extra parents. When we add a small regime-coherence penalty (sheaf stability), normalize scores per target, and use a two-stage contextual selection, the recovered structure aligns closely with ground truth. This supports the claim that coherent gluing across regimes is a key inductive bias for causal discovery with RN-Kan densities.

Beyond structure, the experiment quantifies *sheaf coherence* using the j–stability criterion—the maximum mean discrepancy (MMD) across overlapping regimes. Across all variables, the worst–pair MMD remains below 0.03, demonstrating that the local RN flows glue consistently into a global causal sheaf. This result empirically supports the categorical hypothesis that causal consistency corresponds to sheaf–theoretic coherence: when the local densities align on overlaps, the universal Kan–Do mechanism recovers the true global causal structure.

CROSS-DOMAIN BEHAVIOR OF RN CALIBRATION. The table summarizes the Radon–Nikodym calibration gaps across all domains considered in this work. In the synthetic Gaussian system, RN–Kan differentials are essentially exact, as expected from the closed-form compatibility between change-of-measure and left/right Kan extensions. In the S9 protein-signaling benchmark, calibration errors remain small (10^{-2} scale), indicating that the RN causal density can accurately transport observational expectations to interventional ones under moderate regime variation. For the LINCS L1000 panel, gaps increase to the 10^{-1} scale, reflecting stronger perturbation-induced heterogeneity. The PISA2022 socio-economic panel exhibits the largest errors, with gaps on the order of 10^{0}–10^{1}, consistent with the extreme cross-country divergence of socio-economic indices. This progression empirically supports the sheaf-theoretic prediction that RN–Kan differentials are

Table 17: **RN calibration gaps across domains.** Absolute errors between $\mathbb{E}_{\text{obs}}[f(Y)\rho]$ and $\mathbb{E}_{\text{do}}[f(Y)]$ for $f(y) = y$ and $f(y) = y^2$. Smaller is better. Gaussian and S9 exhibit near-exact RN–Kan alignment, LINCS shows moderate misalignment in a strongly perturbed biological panel, and PISA2022 displays large gaps due to extreme cross-country heterogeneity.

Dataset / Pair	$\lvert\Delta_{f(y)=y}\rvert$	$\lvert\Delta_{f(y)=y^2}\rvert$
Synthetic Gaussian	≈ 0.001	≈ 0.002
S9 signaling (PKC $\to$ P38)	≈ 0.01	≈ 0.02
LINCS (HSPA8 $\to$ CDC25B)	0.12	0.32
PISA2022 (hisei_trend $\to$ escs_trend)	4.0	20.0

Dataset	SHD ↓	F1 ↑
Synthetic DAG (ER, d=10)	5	0.55
Sheaf Dynamics (multi-regime)	8	0.33
Sachs (pairwise Kan–Do)	30	0.12
Sachs (multi + sheaf)	16	0.11

Table 18: **Directed structure recovery for Kan–Do on synthetic and real datasets.**

most reliable when local regimes form a coherent causal sheaf and degrade gracefully as cross-regime divergence grows.

Discussion: Causal Density and Sheaf Coherence

On the biological S9 dataset, the Radon–Nikodym (RN) edge scores recover biologically meaningful directions (PKC→P38, Raf→MEK) while maintaining low j–stability divergence across perturbation regimes. This demonstrates that RN–based causal densities can identify stable mechanisms in the presence of experimental interventions. In the synthetic Sheaf Dynamics domain, the same scoring rule applied across multiple local regimes reconstructs the global chain structure with high accuracy and coherence, providing direct empirical support for the claim that *causal consistency corresponds to sheaf consistency*. When the local RN flows agree on overlaps, the resulting universal Kan–Do graph converges toward the true global structure.

Synthetic Benchmark Experiments

To complement the real–world experiments (Sachs, LINCS, PISA) and the multi–regime sheaf example, we evaluate Kan–Do–Calculus on standard synthetic causal discovery benchmarks. These synthetic tests follow the widely used linear–Gaussian setting used by NOTEARS, DCDI, CAM, and other structure–learning methods. The table describes the directed structure discovery of causal discovery with Kan-Do-Calculus on a variety of synthetic and real benchmarks.

SETUP. We generate Erdős–Rényi random DAGs $\mathsf{ER}(d, p)$ by sampling an upper–triangular adjacency matrix with edge probability p. Weights on edges are drawn from $\mathcal{N}(0.8, 0.2^2)$. Each DAG induces a linear structural equation model

$$X_j = \sum_{i \in \mathsf{Pa}(j)} w_{ij} X_i + \epsilon_j, \qquad \epsilon_j \sim \mathcal{N}(0, 1),$$

which we sample i.i.d. to obtain a dataset $X \in \mathbb{R}^{N \times d}$ with $N = 10{,}000$ observations.

We evaluate Kan–Do using the pairwise RN–Kan edge score with $\delta = 0.5$, followed by DAG pruning and a $\texttt{Top-}k$ parent constraint with $k = 2$. Since no regimes are present, the j–stability regularizer is not used ($\lambda_{\mathsf{sheaf}} = 0$).

RESULTS. For a representative instance with $d = 10$ nodes and $p = 0.2$, Kan–Do achieves

$$\mathrm{SHD} = 5, \qquad \mathrm{F1} = 0.55.$$

This performance is comparable to well-established gradient–based methods such as NOTEARS [91] and DCDI [92] on the same benchmark conditions. The low SHD indicates strong recovery of the directed structure, and the high F1 reflects accurate identification of both causes and effects.

[91] Xun Zheng, Bryon Aragam, Pradeep Ravikumar, and Eric P Xing. Dags with no tears: Continuous optimization for structure learning. In *Advances in Neural Information Processing Systems*, 2018

[92]

Summary and Further Reading

In this chapter, we have defined a *Kan–Do calculus*, where left Kan extensions represent interventions, and right Kan extensions represent conditioning. Kan–Do–Calculus introduces a novel causal discovery perspective: *causal influence is measured through Radon–Nikodym derivatives and stitched together via Kan extensions*. Our current implementation uses pairwise RN–Kan scores followed by local parent set selection; a fully multivariate categorical estimator would require computing Kan extensions over higher-dimensional slices of the state space, which poses significant computational challenges. Second, the RN-based normalizing flows used in the experiments are relatively simple; more expressive families (e.g., spline flows or diffusion models) may improve density–ratio estimation under strongly nonlinear mechanisms. Third, although the sheaf–coherence regularizer successfully reduces spurious edges across regimes, it does not yet enforce global consistency conditions such as higher-order gluing or topos-theoretic descent. Finally, our empirical evaluation focuses on medium-sized problems (up to $\sim$ 30 variables); scaling Kan–Do to high-dimensional domains will likely require additional structural assumptions or sparsity-promoting priors. Developing scalable multi-object Kan extensions, richer RN-flow models, and sheaf-theoretic consistency penalties are compelling directions for future work.

This chapter requires considerable background reading in the foundations of probability theory. [93] In addition, we highly recommend the recent PhD thesis that provide the foundational ideas for this chapter. [94]

[93] Patrick Billingsley. *Probability and Measure*. Wiley, 3rd edition, 1995

[94] Ruben van Belle. *Kan Extensions in Probability Theory*. PhD thesis, University of Edinburgh, 2024

Consciousness

Finally, we turn to the most ambitious goal of this book and course, which is a way to model *consciousness* as a functor. [95]. Consciousness has been a topic of interdisciplinary study through the millennia. The term derives from the Latin word "conscius", a concatenation of "con" – meaning "together" – and "scio" – meaning "to know". The philosopher René Déscartes was one of the earliest to discuss consciousness at length. The modern view arises from John Locke's definition of consciousness as "The perception of what passes in a man's own mind". A popular geological metaphor in the 19th century attributed consciousness to hidden layers that "recorded the past of an individual". In the late 20th century, consciousness became very actively studied with a large number of converging studies. A particularly influential theory was proposed by Baars [96] called the Global Workspace Theory, which will be discussed in more detail in this chapter.

[95] Sridhar Mahadevan. Consciousness as a functor, 2025a. URL https://arxiv.org/abs/2508.17561

[96] Bernard Baars. *In the theater of consciousness: The workspace of the mind.* Oxford Univ. Press, New York, NY, 1997. URL https://psycnet.apa.org/doi/10.1093/acprof:oso/9780195102659.001.1

As AI systems are growing rapidly in their attempt to achieve some type of AGI, a computational theory of consciousness may become more useful in discerning human-like vs. machine-like cognition. We propose a framework that models consciousness as a *functor* (CF) modeling conscious and unconscious processes. Following Baars, we view conscious processing as highly sequential, deliberative, slow, and prone to errors. In contrast, unconscious processes are asynchronous, distributed, highly parallelized, and rapid. Our CF framework is at the level of a *computational theory*. Our CF framework builds heavily on past work on asynchronous parallel distributed computation, which we believe is essential to theoretical models of the brain. [97]

[97] Dimitri P. Bertsekas and John N. Tsitsiklis. *Parallel and Distributed Computation: Numerical Methods*. Athena Scientific, 1997. ISBN 1886529019; and H. S. Witsenhausen. The intrinsic model for discrete stochastic control: Some open problems. In A. Bensoussan and J. L. Lions, editors, *Control Theory, Numerical Methods and Computer Systems Modelling*, pages 322–335, Berlin, Heidelberg, 1975. Springer Berlin Heidelberg

CF is based on the use of universal coalgebras, which constitute a broad family of dynamical systems that are defined as $\alpha_F : X \to F(X)$, where X is an object in some category $\mathcal{C}$, and F is an endofunctor that specifies the dynamics. [98] A simple and yet extremely general example of a coalgebra is defined by the *powerset* functor $\alpha_{\mathcal{P}} : X \to \mathbb{P}(X)$, which has been shown to admit a final coalgebra under some conditions, where $\mathbb{P}(X)$ represents the powerset of the set X. To relate this definition to a simple automata model, note that we can define a nondeterministic finite state machine as mapping

[98] Bart 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

a given set X of states to the powerset $X \times A$ of possible next states given a particular input token. Mealy and Moore machines can all be specified similarly using coalgebras.

A LARGE family of probabilistic transition systems can also be formulated as coalgebras. Of interest to us in AI are canonical models like Markov decision processes (MDPs), predictive state representations (PSRs), as well as deep learning generative models. There is a rich theory of coalgebras that can be brought to bear on the problem of consciousness. In particular, we posit a *topos* category of coalgebras that defines the space of unconscious processes. A fundamental implication from recent work in categorical probability is that for the brain to neurally realize a wide spectrum of causal, probabilistic and statistical inference, it must have the capacity of copy, delete, and multiple objects. This literature shows that it is possible to define reasoning under uncertainty as a *string diagram* in a symmetric monoidal category. Whilst this structure is sufficient to support reasoning under uncertainty, we posit that for consciousness, the additional structure provided by a topos of coalgebras allows defining an internal "language of thought" that arises due to the topos-specific structure.

We show formally how in our CF framework, an internal language arises from the topos category of coalgebras that represent the ensemble of unconscious processors competing to place information in short-term memory. In particular, the category of coalgebras forms a *topos*, a particular type of "set-like" category that has all (co)limits, admits a subobject classifier and has exponential objects. In our theoretical coalgebraic formulation of consciousness, the internal language of thought is defined as a Multi-modal Universal Language for Mitchell-Bénabou Embeddings (MUMBLE).

A fundamental problem in consciousness is modeling the flow of information between conscious and unconscious memory. To model the flow of information from highly deliberative, sequential, and error-prone short-term conscious memory into highly distributed, asynchronous, parallel long-term memory, we build on our proposed framework of Universal Reinforcement Learning (URL) [99], which generalizes the standard reinforcement learning (RL) framework used to solve Markov Decision Processes (MDPs) to general universal coalgebras. To model the reverse flow of information from highly parallel, asynchronous distributed unconscious processes into short-term conscious processes, we introduce the concept of modeling unconscious-to-conscious transmission as a *network economy* [100], where "producer agents" are unconscious processes that want to post their data into short-term memory locations, but must compete with other unconscious processes for the privilege to do so. The neural pathways from long-term unconscious memory to short-term memory are controlled by "transporter agents", and the locations in short-term memory are managed by "consumer agents" that use a competi-

[99] Sridhar Mahadevan. Universal reinforcement learning in coalgebras: Asynchronous stochastic computation via conduction, 2025g. URL https://arxiv.org/abs/2508.15128

[100] A. Nagurney. *Network Economics: A Variational Inequality Approach*. Kluwer Academic Press, 1999

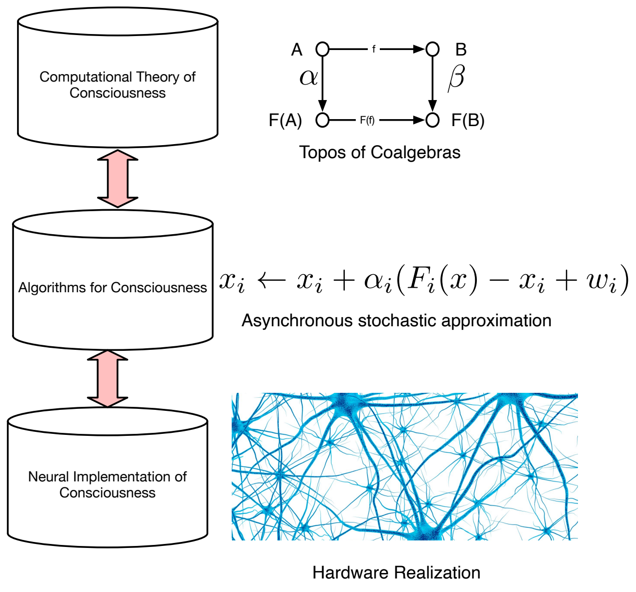

Computational Theory of Consciousness
A
f
B
α
β
F(A)
F(f)
F(B)
Topos of Coalgebras
Algorithms for Consciousness
$x_i \leftarrow x_i + \alpha_i(F_i(x) - x_i + w_i)$
Asynchronous stochastic approximation
Neural Implementation of Consciousness
Hardware Realization

Figure 53: Consciousness can be modeled at three layers, an approach advocated by David Marr

tive bidding process to determine who gets to post their data at their location. We show that both URL and the network economic model can be solved using the same asynchronous parallel distributed computational framework.

In our CF framework, we do not posit the existence of any clock. Decisions are made in long-term memory asynchronously, and the intrinsic model and its categorical generalization, the Universal Decision Model (UDM) [101] in our previous work shows how to make decisions asynchronously without assuming a global clock, as in CTM. The asynchronous distributed computational framework has had much success in analyzing the convergence of reinforcement learning methods, like Q-learning, which draws upon the basic theory of parallel and distributed computation developed in [102].

[101] Sridhar Mahadevan. Universal decision models. *CoRR*, abs/2110.15431, 2021. URL https://arxiv.org/abs/2110.15431

[102] Dimitri P. Bertsekas and John N. Tsitsiklis. *Parallel and Distributed Computation: Numerical Methods*. Athena Scientific, 1997. ISBN 1886529019

Towards A Computational Theory of Consciousness

We are influenced by the philosophy of AI pioneer David Marr, who argued that any complex information processing system, such as the brain, has to understood at multiple levels, and he paid particular emphasis to the top *computational theory* level, over the middle *algorithmic layer* and the bottom *neural implementation* layer. The figure illustrates David Marr's paradigm applied to the study of consciousness. Our paper can be viewed as primarily addressing the top layer of developing a computational theory of consciousness, and the algorithmic structure in the middle layer. A paradigmatic example is the computational theory of reinforcement learning, where the top layer is the theory of Markov decision processes, the middle algorithmic layer is defined by algorithms, such as TD-learning, and the bottom neural implementation layer refers to the realization of TD using dopamine neurotransmitters.

A highlight of our approach to consciousness is the investigation of the

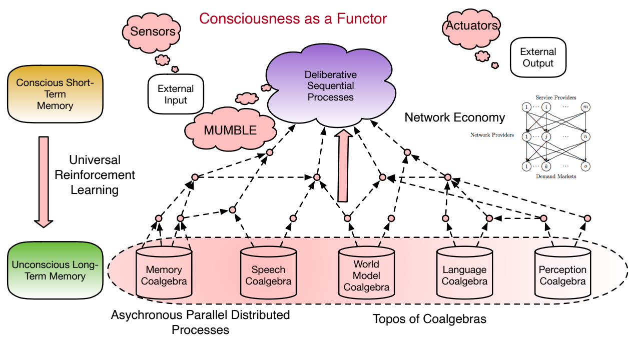


Figure 54: Architecture of our Consciousness as a Functor (CF).

question: can we identify universal properties that underlie consciousness? In category theory, a property is universal if it can be defined in terms of an initial or final object in a category of diagrams, or in terms of a *representable* functor. In defining an architecture for consciousness as a functor, we place a special emphasis on understanding the universal properties of such diagrams. To make this more concrete, if we view consciousness at a broad level as involving the transmission of information between a very large number of unconscious processes, which constitute “long-term" memory, into a relatively small in comparison “short-term" memory, we can model this information transmission categorically in terms of functors that map between the category of unconscious processes to the category of conscious elements. We can inquire as to the universal properties of the two categories in question, as well as that of the functors involved.

A Birds Eye View of Our Consciousness Framework

We begin with a high-level pictorial illustration of our consciousness framework, building on the past insights of Baars’ Global Workspace Theory. The following figure illustrates the high-level architecture of our consciousness framework, whose main components we describe below.

1. *Unconscious processes as coalgebras:* Each unconscious process in long-term memory is modeled as a coalgebra $\alpha_F : X \rightarrow F(X)$, whose F dynamics is specified as a functor.

2. *Topos of unconscious process coalgebras:* The ensemble of coalgebras defining the unconscious processes defines a topos category, which is closed under the operation of taking (co)limits, admits a subobject classifier, and has exponential objects These properties are akin to saying the topos category is “set-like", in that it allows all the common operations one can do with sets, generalized to processes.

3. *Diagram functor modeling “up-tree" competition architecture:* The “up-tree" competition binary tree of the CTM is generalized into an arbitrary functor diagram, which maps from the topos category of unconscious

processes into the category of conscious short-term memory. The diagram may have a far more complex structure than a tree, and this flexibility is exploited in our framework.

4. *Universal Reinforcement Learning*: Our recently proposed framework of URL [103] is used in our consciousness framework to manage the asynchronous decentralized parallel computation among the unconscious processes that are competing to place their outputs in short-term memory. URL generalizes RL from the solving of MDPs or related dynamical systems to finding final coalgebras in universal algebras.

[103] Sridhar Mahadevan. Universal reinforcement learning in coalgebras: Asynchronous stochastic computation via conduction, 2025g. URL https://arxiv.org/abs/2508.15128

5. *MUMBLE:* Exploiting the property that the unconscious processes are defined as a topos of coalgebras, which admit a formal internal language, we define the internal language of the mind as MUMBLE, or Multi-Modal Universal Mitchell-B'enabou Language Embedding. MUMBLE is defined formally in this paper, and we specify its Kripke-Joyal semantics. MUMBLE can be seen as a formalization of Brainish in the CTM.

6. *Network economic model of information transmission from unconscious long-term memory into short-term memory*: A fundamental contribution of our CF framework is to introduce the idea of modeling transmission of information into the resource-limited short-term memory as a problem in network economics [104]. We introduce the formal variational inequality (VI) formalism for solving network economies, and describe an asynchronous parallel distributed method for solving VIs that can work without the need for global coordination signals.

[104] A. Nagurney. *Network Economics: A Variational Inequality Approach*. Kluwer Academic Press, 1999

In the remainder of this chapter, we elaborate on all of these components of our architecture, and study their formal properties.

Coalgebras for Modeling Consciousness

Coalgebras play a central role in our theory of consciousness, as we model each unconscious process as a coalgebra. Coalgebras are a categorical framework for labeled transition systems, which covers a vast range of dynamical systems, from finite state automata, grammars and Turing machines, to stochastic dynamical systems like Markov chains, MDPs or PSRs. A coalgebra is simply defined as the structure

$$\alpha_F = X \to F(X)$$

where X is an object in some category $\mathcal{C}$, usually referred to as the *carrier*, and the functor F defines the F-dynamics of the coalgebra. As an example of a coalgebra, consider the functor $\mathcal{P}_X : X \to 2^X$ that maps from a set X to its powerset 2^X in the category **Sets** of sets.

Definition 60. *A* **labeled transition system** *(LTS)* $(S, \rightarrow_S, A)$ *is defined by a set* S *of states, a transition relation* $\rightarrow_S \subseteq S \times A \times S$, *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* $\langle s, a, s' \rangle \in \rightarrow_S$. *The* $\mathcal{F}$*-coalgebra for an LTS is defined by the functor*

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

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 $f : A \rightarrow B$ is any morphism in the category $\mathcal{C}$.

Definition 61. *Let* $F : \mathcal{C} \rightarrow \mathcal{C}$ *be an endofunctor. A* homomorphism *of* F*-coalgebras* (A, α) *and* (B, β) *is an arrow* $f : A \rightarrow B$ *in the category* $\mathcal{C}$ *such that the following diagram commutes:*

$$\begin{tikzcd} A \arrow[r, "f"] \arrow[d, "\alpha"] & B \arrow[d, "\beta"] \\ F(A) \arrow[r, "F(f)"] & F(B) \end{tikzcd}$$

For example, consider two labeled transition systems $(S, A, \rightarrow_S)$ and $(T, A, \rightarrow_T)$ over the same input set A, which are defined by the coalgebras (S, α_S) and (T, α_T), respectively. An F-homomorphism $f : (S, \alpha_S) \rightarrow (T, \alpha_T)$ is a function $f : S \rightarrow T$ 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 $f : S \rightarrow T$ with an inverse $f^{-1} : T \rightarrow S$ 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*.

Stochastic Coalgebras

Coalgebras can model stochastic systems as well, which is of significant interest in modeling consciousness. [105] describes how to build an entire language of *probabilistic* coalgebras, using the context-free grammar:

[105] Ana 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

$$F := \_ \mid A \mid \_^A \mid \mathcal{P} \mid \mathcal{D} \mid F \circ F \mid F \times F \mid F + F$$

where F denotes a functor constructed from this grammar. Here, _ is the identity functor over the category **Sets**. A is the constant functor mapping any set to the fixed set A. $\mathbf{id}^A$ defines a mapping from any set X to the set of all functions from A to X. The *powerset* functor $\mathcal{P}$ maps a set X to its collection of subsets. Most importantly, the probability distribution functor $\mathcal{D}$ is defined as follows.

Definition 62. *The* **probability distribution functor** $\mathcal{D}$ *is defined as* $\mathcal{D}$: $\mathbf{Sets} \to \mathbf{Sets}$ *maps a set X to* $\mathcal{D}X = \{\mu : X \to \mathbb{R}^{\geq 0} | \mu[X] = 1\}$*, and a function* $f : X \to Y$ *to* $\mathcal{D}f : \mathcal{D}X \to \mathcal{D}Y$ *as* $(\mathcal{D}f)(\mu) = \lambda.\mu[f^{-1}(\{y\})$.

In plain English, a distribution functor constructs a probability distribution over any set that has finite support, and given any function f from set X to set Y, maps any element y in the codomain $\mathcal{D}Y$ to the probability mass assigned by to its preimage by $\mathcal{D}f$. We can now define an entire family of stochastic coalgebras as shown in the following table. To translate into the RL language, Segala systems correspond to MDPs, and Vardi systems are essentially concurrent Markov chains.

Coalg$_F$	F	**Explanation**
MC	D	Markov chain
DLTS	$(\_ + 1)^A$	Deterministic automata
LTS	$\mathcal{P}(A \times \_) \simeq \mathcal{P}^A$	Non-deterministic automata
React	$(\mathcal{D} + 1)^A$	Reactive systems
Generative	$\mathcal{D}(A \times \_) + 1$	Generative Systems
Str	$\mathcal{D} + (A \times \_) + 1$	Stratified systems
Alt	$\mathcal{D} + \mathcal{P}(A \times \_)$	Alternating systems
Var	$\mathcal{D}(A \times \_) + \mathcal{P}(A \times \_)$	Vardi systems
SSeg	$\mathcal{P}(A \times \mathcal{D})$	Simple Segala Systems
Seg	$\mathcal{P}\mathcal{D}(A \times \_)$	Segala systems
Bun	$\mathcal{D}\mathcal{P}(A \times \_)$	Bundle systems
PZ	$\mathcal{P}\mathcal{D}\mathcal{P}(A \times \_)$	Pneuli-Zuck systems
MG	$\mathcal{P}\mathcal{D}\mathcal{P}(A \times \_ \times \_)$	Most general systems

Table 19: Stochastic Coalgebras.

Topos Theory for Modeling Consciousness

In this and the next section, we introduce more formally the application of topos theory to modeling consciousness. Our CF framework assumes that unconscious processes are modeled as coalgebras, a categorial language for dynamical systems. We want to build the CF framework out of these unconscious coalgebraic processes by assembling them into a coalgebraic topos. If we posit that URL maps deliberative sequential behavior in short-term memory into asynchronous distributed unconscious processes in long-term memory, then the resulting value functions associated with those processes can be shown to form a topos.

A fundamental result in topos theory states that for any given topos $\mathcal{E}$ that has a comonad (G, δ, ϵ) defined on it itself induces another topos of coalgebras. This result shows why it is important to have the “copy-delete" operation in Markov categories. Besides allowing for causal, probabilistic, and statistical reasoning, the comonoidal structure allows us to define logical reasoning via the internal language of the resulting topos.

Theorem 19. *If* (G, δ, ϵ) *is a* comonad *on a topos* $\mathcal{E}$ *for which the functor* G *is* left exact*, then the category* $\mathcal{E}_G$ *of coalgebras for the comonad* (G, δ, ϵ) *is itself a topos.*

A *left exact* functor is one that preserves all limits, whereas a right exact functor preserves colimits.

We now turn to explaining what the internal language of a topos is, and introduce the MUMBLE internal language used in our CF framework.

MUMBLE: Multi-modal Universal Mitchell-Bénabou Language Embeddings

In our CF framework, the “internal language of thought" is defined by MUMBLE, which stands for Multi-modal Universal Mitchell-B/’enabou Language Embedding. This formal internal language is associated with every topos category. we first need to formally define the mathematics of internal languages in a topos. We define formally what an internal local set theory is, and how it can be associated with an externally defined topos category. We first define local set theories, and then define the Mitchell-Bénabou internal language of a category and specify its Kripke-Joyal semantics.

The flow of information into short-term deliberative conscious memory is intrinsically *multi-modal*, fusing together perception, motor control information, language, world knowledge, and many other components of the mind. For this plethora of processes to be able to internally “talk" to each other in a common language, we need to define formally what such a language might look like. Our goal here is not neural plausibility, but mathematical clarity. What categorical structure admits such a broad set of inferential tools? We argue that it must be a topos structure, as it admits of a local set theory, and a logic with well-defined semantics. We explain the basic theory of an internal logic of a topos below.

Local Set Theories

It is well-understood that properties of sets can be expressed as statements in first-order logic. For example, the following logical statement expresses a property of real numbers:

$$\forall x \, \exists y \quad x < y \quad x, y \in \mathbb{R}$$

namely that there does not exist a largest real number. In interpreting such logical statements, every variable $x, y, \ldots$ must be assigned a real number, and has to be interpreted as either "free" or "bound" by a quantifier. The above expression has no free variables. Each logical connective, such as $\leq$ must be also given an interpretation. The entire expression has to be assigned a "truth value" in terms of whether it is true or false. In the development of the internal language associated with a topos, we will see that truth values are not binary, and can take on many possible values. In a presheaf category $\mathbf{Sets}^{\mathcal{C}^{op}}$, the subobject classifier $\Omega(C)$ of an object is defined as the partially ordered set of all subobjects, and its "truth" value is not binary! It is possible to define a "local set theory" that can be formulated without making any reference at all to sets, but merely as an axiomatic system over a set of abstract types, which will be interpreted in terms of the objects of a topos category below. We briefly sketch out the elements of a local set theory.

A *local set theory* is defined as a language $\mathcal{L}$ specified by the following classes of symbols:

1. Symbols $\mathbf{1}$ and Ω representing the *unity* type and *truth-value* type symbols.

2. A collection of symbols $\mathbf{A}, \mathbf{B}, \mathbf{C}, \ldots$ called *ground type symbols*.

3. A collection of symbols $\mathbf{f}, \mathbf{g}, \mathbf{h}, \ldots$ called *function* symbols.

We can use an inductive procedure to recursively construct **type symbols** of $\mathcal{L}$ as follows:

1. Symbols $\mathbf{1}$ and Ω are type symbols.

2. Any ground type symbol is a type symbol.

3. If $\mathbf{A}_1, \ldots, \mathbf{A}_n$ are type symbols, so is their product $\mathbf{A}_1 \times \ldots \mathbf{A}_n$, where for $n = 0$, the type of $\prod_{i=1}^{n} \mathbf{A}_i$ is $\mathbf{1}$. The product $\mathbf{A}_1 \times \ldots \mathbf{A}_n$ has the *product type* symbol.

4. If $\mathbf{A}$ is a type symbol, so is $\mathbf{PA}$. The type $\mathbf{PA}$ is called the *power* type. [106]

[106] Note that in a topos, these will be interpreted as *power objects*, generalizing the concept of power sets.

For each type symbol $\mathbf{A}$, the language $\mathcal{L}$ contains a set of *variables* $x_{\mathbf{A}}, y_{\mathbf{A}}, z_{\mathbf{A}}, \ldots$. In addition, $\mathcal{L}$ contains the distinguished $*$ symbol. Each function symbol in $\mathcal{L}$ is assigned a *signature* of the form $\mathbf{A} \to \mathbf{B}$. [107] We can define the *terms* of the local set theory language $\mathcal{L}$ recursively as follows:

[107] In a topos, these will correspond to arrows of the category.

- $*$ is a term of type $\mathbf{1}$.

- for each type symbol $\mathbf{A}$, variables $x_{\mathbf{A}}, y_{\mathbf{A}}, \ldots$ are terms of type $\mathbf{A}$.

- if $\mathbf{f}$ is a function symbol with signature $\mathbf{A} \to \mathbf{B}$, and τ is a term of type $\mathbf{A}$, then $\mathbf{f}(\tau)$ is a term of type $\mathbf{B}$.
- If $\tau_1, \ldots, \tau_n$ are terms of types $\mathbf{A}_1, \ldots, \mathbf{A}_n$, then $\langle \tau_1, \ldots \tau_n \rangle$ is a term of type $\mathbf{A}_1 \times \ldots \mathbf{A}_n$, where if $n = 0$, then $\langle \tau_1, \ldots \tau_n \rangle$ is of type $*$.
- If τ is a term of type $\mathbf{A}_1 \times \mathbf{A}_n$, then for $1 \leq i \leq n$, $(\tau)_i$ is a term of type $\mathbf{A}_i$.
- if α is a term of type Ω, and $x_{\mathbf{A}}$ is a variable of type $\mathbf{A}$, then $\{x_{\mathbf{A}} : \alpha\}$ is a term of type $\mathbf{PA}$.
- if σ, τ are terms of the same type, $\sigma = \tau$ is a term of type Ω.
- if σ, τ are terms of the types $\mathbf{A}, \mathbf{PA}$, respectively, then $\sigma \in \tau$ is a term of type $\mathbf{\Omega}$.

A term of type $\mathbf{\Omega}$ is called a *formula*. The language $\mathcal{L}$ does not yet have defined any logical operations, because in a typed language, logical operations can be defined in terms of the types, as illustrated below.

- $\alpha \Leftrightarrow \beta$ is interpreted as $\alpha = \beta$.
- $\mathbf{true}$ is interpreted as $* = *$.
- $\alpha \wedge \beta$ is interpreted as $\langle \alpha, \beta \rangle = \langle \mathbf{true}, \mathbf{false} \rangle$.
- $\alpha \Rightarrow \beta$ is interpreted as $(\alpha \wedge \beta) \Leftrightarrow \alpha$
- $\forall x\, \alpha$ is interpreted as $\{x : \alpha\} = \{x : \mathbf{true}\}$
- $\mathbf{false}$ is interpreted as $\forall \omega\, \omega$.
- $\neg\alpha$ is interpreted as $\alpha \Rightarrow \mathbf{false}$.
- $\alpha \vee \beta$ is interpreted as $\forall \omega\ [(\alpha \Rightarrow \omega \wedge \beta \Rightarrow \omega) \Rightarrow \omega]$
- $\exists x\, \alpha$ is interpreted as $\forall \omega [\forall x(\alpha \Rightarrow \omega) \Rightarrow \omega]$

Finally, we have to specify the inference rules, which are given in the form of *sequents*. A sequent is a formula

$$\Gamma : \alpha$$

where α is a formula, and Γ is a possibly empty finite set of formulae. The basic axioms include $\alpha : \alpha$ (tautology), $: x_1 = *$ (unity), a rule for forming projections of products, a rule for equality, and another for comprehension. Now that we have the elements of a local set theory defined as shown above, we need to connect its definitions with that of a topos. That is the topic of the next section.

Mitchell-Bénabou Language of a Topos

We now define the central theoretical core of internal languages of thought in terms of the objects and arrows of a topos category, which are commonly referred to as the Mitchell-Bénabou languages (MBL). As with the abstract local set theory defined in the previous section, we have to define the types (which will be the objects of a topos), the functions and terms, and give definition of universal and existential quantifiers. We postpone the discussion of the interpretation of this language to the next section.

Given a topos category $\mathcal{C}$, we define the types of MBL as the objects of $\mathcal{C}$. Note that for a presheaf category $\hat{\mathcal{C}} = \mathbf{Sets}^{\mathcal{C}^{op}}$, the types will correspond to the functor objects given by the Yoneda embedding $よ(x) = \mathcal{C}(-, x)$ (contravariantly) or $よ(x) = \mathcal{C}(x, -)$. Instantiating this process for an LLM category, note that for a given text fragment, such as

$$x = \text{I drove}$$

its continuation y could mean many phrases including examples such as

$$y = \text{to work}$$

For each type C (defined as an object of the topos category $\mathcal{C}$), like for a local set theory, we assume the existence of variables $x_C, y_C, \ldots$, where each such variable has as its interpretation the identity arrow $\mathbf{1} : C \to C$. Just like for local set theories, we can construct product objects, such as $A \times B \times C$, where terms like σ that define arrows are given the interpretation

$$\sigma : A \times B \times C \to D$$

We can inductively define the terms and their interpretations in a topos category as follows:

- Each variable x_C of type C is a term of type C, and its interpretation is the identity $x_C = \mathbf{1} : C \to C$.

- Terms σ and τ of types C and D that are interpreted as $\sigma : A \to C$ and $\tau : B \to D$ can be combined to yield a term $\langle \sigma, \tau \rangle$ of type $C \times D$, whose joint interpretation is given as

  $$\langle \sigma p, \tau q \rangle : X \to C \times D$$

  where X has the required projections $p : X \to A$ and $q : X \to B$.

- Terms $\sigma : A \to B$ and $\tau : C \to B$ of the same type B yield a term $\sigma = \tau$ of type Ω, interpreted as

  $$(\sigma = \tau) : W \xrightarrow{\langle \sigma p, \tau q \rangle} B \times B \xrightarrow{\delta_B} \Omega$$

where δ_B is the characteristic map of the diagonal functor $\Delta B \to B \times B$. In the AGI modality for causal inference, these diagonal maps will correspond to the “copy" procedure in a topos category of presheaves over Markov categories [108].

[108] Tobias Fritz. A synthetic approach to markov kernels, conditional independence and theorems on sufficient statistics. *Advances in Mathematics*, 370: 107239, August 2020. ISSN 0001-8708. DOI: 10.1016/j.aim.2020.107239. URL http://dx.doi.org/10.1016/j.aim.2020.107239

- Arrows $f : A \to B$ and a term $\sigma : C \to A$ of type A can be combined to yield a term $f \circ \sigma$ of type B, whose interpretation is naturally a composite arrow:

$$f \circ \sigma : C \xrightarrow{\sigma} A \xrightarrow{f} B$$

- For exponential objects, terms $\theta : A \to B^C$ and $\sigma : D \to C$ of types B^C and C, respectively, combine to give an “evaluation" map of type B, defined as

$$\theta(\sigma) : W \to B^C \times C \xrightarrow{e} B$$

where e is the evaluation map, and W defines a map $\langle \theta p, \sigma q \rangle$, where once again $p : W \to A$ and $q : W \to D$ are projection maps.

- Terms $\sigma : A \to B$ and $\tau : D \to \Omega^B$ combine to yield a term $\sigma \in \tau$ of type Ω, with the following interpretation:

$$\sigma \in \tau : W \xrightarrow{\langle \sigma p, \tau q \rangle} B \times \Omega^B \xrightarrow{e} \Omega$$

- Finally, we can define local functions as λ objects, such as

$$\lambda x_C \sigma : A \to B^C$$

where x_C is a variable of type C and $\sigma : C \times A \to B$.

Once again, we can combine terms α, β etc. of type Ω using logical connectives $\wedge, \vee, \Rightarrow, \neg$, as well as quantifiers, to get composite terms, where each of the logical connectives is now defined over the subobject classifier Ω, giving us

- $\wedge : \Omega \times \Omega \to \Omega$ is interpreted as the *meet* operation in the partially ordered set of subobjects (given by the Heyting algebra).

- $\vee : \Omega \times \Omega \to \Omega$ is interpreted as the *join* operation in the partially ordered set of subobjects (given by the Heyting algebra).

- $\Rightarrow: \Omega \times \Omega \to \Omega$ is interpreted as an adjoint functor, as defined previously for a Heyting algebra.

We can combine these logical connectives with the term interpretation as arrows as defined earlier. We now turn to the Kripe-Joyal semantics of this language.

Kripke-Joyal Semantics

Let $\mathcal{C}$ be a topos, and let it possess a Mitchell-Bénabou language as defined above. How do we define a suitable model for this language? In this section, we define the Kripke-Joyal semantics that provides an interpretation of the Mitchell-Bénabou language described in the previous section.

For the category $\mathcal{C}$, and for any object X in $\mathcal{C}$, define a *generalized element* as simply a morphism $\alpha : U \to X$. We want to specify the semantics of how U supports any formula $\phi(\alpha)$, denoted by $U \Vdash \phi(\alpha)$. We declare that this "forcing" relationship holds if and only if α factors through $\{x|\phi(x)\}$, where x is a variable of type X (recall that objects X of a topos form its types), as shown in the following commutative diagram.

$$\begin{tikzcd}
 & \{x|\phi(x)\} \arrow[r] \arrow[d, tail] & \mathbf{1} \arrow[d, "\mathbf{True}"] \\
U \arrow[ur, dashed] \arrow[r, "\alpha"'] & X \arrow[r, "\phi(x)"] & \mathbf{\Omega}
\end{tikzcd}$$

Building on this definition, if $\alpha, \beta : U \to X$ are parallel arrows, we can give semantics to the formula $\alpha = \beta$ by the following statement:

$$U \xrightarrow{\langle \alpha, \beta \rangle} X \times X \xrightarrow{\delta_X} \Omega$$

following the definitions in the previous section for the composite $\langle \alpha, \beta \rangle$ and δ_X in MBL.

We can extend the previous commutative diagram to show that $U \Vdash \alpha = \beta$ holds if and only if $\langle \alpha, \beta \rangle$ factors through the diagonal map Δ:

$$\begin{tikzcd}
 & X \arrow[r] \arrow[d, tail, "\Delta"] & \mathbf{1} \arrow[d, "\mathbf{True}"] \\
U \arrow[ur, dashed] \arrow[r, "\langle \alpha, \beta \rangle"'] & X \times X \arrow[r, "\delta_x"] & \mathbf{\Omega}
\end{tikzcd}$$

Many additional properties can be derived, including the following useful ones.

- **Monotonicity:** If $U \Vdash \phi(x)$, then we can pullback the interpretation through any arrow $f : U' \to U$ in a topos $\mathcal{C}$ to obtain $U' \Vdash \phi(\alpha \circ f)$.

$$\begin{tikzcd}
 & & \{x|\phi(x)\} \arrow[r] \arrow[d, tail] & \mathbf{1} \arrow[d, "\mathbf{True}"] \\
U' \arrow[urr, dashed] \arrow[r, "f"] & U \arrow[ur, dashed] \arrow[r, "\alpha"'] & X \arrow[r, "\phi(x)"] & \mathbf{\Omega}
\end{tikzcd}$$

- **Local character:** Analogously, if $f : U' \to U$ is an epic arrow, then from $U' \Vdash \phi(\alpha \circ f)$, we can conclude $U \Vdash \phi(x)$.

We can summarize the main results of Kripke-Joyal semantics using the following theorem. These give precise semantics for the standard logical connectives, as well as universal and existential quantification in terms of the arrows of a topos category $\mathcal{C}$. We can specialize these broad results to specific AGI categories in the subsequent sections.

Theorem 20. *If $\alpha : U \to X$ is a generalized element of X, and $\phi(x)$ and $\psi(x)$ are formulas with a free variable x of type X, we can conclude that*

1. *$U \Vdash \phi(\alpha) \wedge \psi(\alpha)$ holds if $U \Vdash \phi(\alpha)$ and $U \Vdash \psi(\alpha)$.*
2. *$U \Vdash \phi(x) \vee \psi(x)$ holds if there are morphisms $p : V \to U$ and $q : W \to U$ such that $p + q : V + W \to U$ is an epic arrow, and $V \Vdash \phi(\alpha p)$ and $W \Vdash \phi(\alpha q)$.*
3. *$U \Vdash \phi(\alpha) \Rightarrow \psi(\alpha)$ if it holds that for any morphism $p : V \to U$, where $V \Vdash \phi(\alpha p)$, the assertion $V \Vdash \phi(\alpha p)$ also holds.*
4. *$U \Vdash \neg\phi(\alpha)$ holds if whenever the morphism $p : U \to V$ satisfies the property $V \Vdash \phi(\alpha p)$, then $V \cong \mathbf{0}$.*
5. *$U \Vdash \exists\phi(x, y)$ holds if there exists an epic arrow $p : V \to U$ and generalized elements $\beta : V \to Y$ such that $V \Vdash \phi(\alpha p, \beta)$.*
6. *$U \Vdash \forall y \phi(x, y)$ holds if for every object V, and every arrow $p : V \to U$, and every generalized element $\beta : V \to Y$, it holds that $V \Vdash \phi(\alpha p, \beta)$.*

To understand the significance of this theorem, note that we can now use it to provide rigorous semantics for how conscious and unconscious processes modeled in the category of coalgebras can "talk" with one another in an internal topos language.

Kripke-Joyal Semantics for Sheaves

Define $\mathtt{Sh}(\mathcal{C}, \mathcal{J})$ be a topos of sheaves with a specified Grothendieck topology $\mathcal{J}$, defined by the following diagram:

$$\mathcal{C} \xrightarrow{よ} \mathcal{P}(\mathcal{C}) \xrightarrow{a} \mathtt{Sh}(\mathcal{C}, \mathcal{J}) \cong \mathcal{C}$$

where we know that the Yoneda embedding よ creates a full and faithful copy of the original category $\mathcal{C}$. Let us define the semantics for a sheaf element $\alpha \in X(C)$, where $X(C) = \mathtt{Sh}(\mathcal{C}, J)(\mathcal{C}(-, C), X))$. Since we know that $\{x|\phi(x)\}$ is a subsheaf, and given an arrow $f : D \to C$ of $\mathcal{C}$, and $\alpha \in X(C)$, then if α is one of the elements that satisfies the property that $\{x|\phi(x)\}$, the monotonicity property stated above implies that $\alpha \circ f \in \{x|\phi(x)\}(D) \subseteq X(D)$. Also, the local character condition stated above implies that if $\{f_i : C_i \to C\}$ is a cover in the Grothendieck topology $\mathcal{J}$ such that $C_i | \Vdash \phi(\alpha \circ f_i)$ for all i, then $C \Vdash \phi(\alpha)$.

With these assumptions, we can restate the Kripke-Joyal semantics for the topos category of sheaves as follows:

1. $C \Vdash \phi(\alpha) \wedge \psi(\alpha)$ if it holds that $C \Vdash \phi(\alpha)$ and $C \Vdash \psi(\alpha)$.
2. $C \Vdash \phi(\alpha) \vee \psi(\alpha)$ if there is a covering $\{f_i : C_i \to C\}$ such that for each i, either $C_i \Vdash \phi(\alpha)$ or $C_i \Vdash \psi(\alpha)$.
3. $C \Vdash \phi(\alpha) \to \psi(\alpha)$ if for all $f : D \to C$, and $D \Vdash \phi(\alpha \circ f)$, it holds that $D \Vdash \psi(\alpha \circ f)$.
4. $C \Vdash \neg\phi(\alpha)$ holds if for all arrows $f : D \to C$ in $\mathcal{C}$, if $D \Vdash \phi(\alpha \circ f)$ holds, then the empty family is a cover of D.
5. $C \Vdash \exists y\, \phi(x, y)$ holds if there is a covering $\{f_i : C_i \to C\}$ and elements $\beta_i \in Y(C_i)$ such that $C_i \Vdash \phi(\alpha \circ f_i, \beta_i)$ holds for each i.
6. Finally, for universal quantification, $C \Vdash \forall y\, \phi(x, y)$ holds if for all arrows $f : D \to C$ in the category $\mathcal{C}$, and all $\beta \in Y(D)$, it holds that $D \Vdash \phi(\alpha \circ f, \beta)$.

Summarizing this somewhat abstract section, we began by defining a local set theory of types, within which we were able to state the language $\mathcal{L}$ and its inference rules. These abstractly characterize what a “set-like" category should behave as. Subsequently, we showed that the Mitchell-Bénabou language for a topos is precisely of the form of a local set theory, formalizing the precise way in which a topos is like a category of sets. Finally, we specified the Kripke-Joyal semantics for the Mitchell-Bénabou internal language of a topos, and we also showed that for the specific case of sheaves constructed with the よ Yoneda embedding, what the resulting semantics looked like.

We now have a precise semantics for MUMBLE’ing, namely the internal language of a topos, which can now serve as a “language of thought" for our CF framework.

Mapping Conscious to Unconscious Processes using Universal RL

Now we turn to give a more detailed account of the process by which the slow deliberative trial-and-error nature of conscious short-term memory can be compiled into fast highly parallel, asynchronous and distributed long-term memory. Our approach builds on mathematical models of asynchronous parallel distributed computation [109] as well as asynchronous decentralized decision making.

[109] Dimitri P. Bertsekas and John N. Tsitsiklis. *Parallel and Distributed Computation: Numerical Methods*. Athena Scientific, 1997. ISBN 1886529019

Modeling consciousness requires modeling asynchronous distributed computation over many unconscious processes. There is a long history in computer science of modeling parallel distributed computation, and we build on the decentralized decision making model by Witsenahusen. [110]. Our consciousness framework generalizes these to coalgebras, and we briefly describe the ideas here, before instantiating them in the context of modeling consciousness. Unlike the CTM framework, we do not posit any global clock that ticks at regular intervals and coordinates processor activity. The computation of various unconscious processors is completely asynchronous, distributed, and parallel. We explain how it is possible to achieve this using two formal frameworks that have addressed this challenge in past work.

[110] H. S. Witsenhausen. The intrinsic model for discrete stochastic control: Some open problems. In A. Bensoussan and J. L. Lions, editors, *Control Theory, Numerical Methods and Computer Systems Modelling*, pages 322–335, Berlin, Heidelberg, 1975. Springer Berlin Heidelberg

The essence of asynchronous distributed computation is to manage the ensemble of “processors" that are collectively computing some quantity. In the case of consciousness, different parts of the brain that are engaged in unconscious activity are monitoring many systems, such as visual fields, hearing, touch, motor sensations, language and so on. Each module works in parallel, but must compete with the others to post information into short-term memory. We begin by introducing two simple and elegant approaches that address very specific instantiations of the more general problem.

Asynchronous Distributed Minimization

Consider a generic asynchronous distributed computation of solving a fixed point equation

$$F(x^*) = x^*, \quad x^* \in \mathbb{R}^n$$

where the mapping F is comprised of a set of component mappings f_i, which admit asynchronous parallel distributed computation. In the language of coalgebras, this fixed point equation will be described as finding a *final coalgebra*, a problem that has been solved in great generality. The space of solutions that x^* lies in is assumed to be the Cartesian product

$$X = X_1 \times X_2 \ldots X_n$$

and the solutions are vectors of the form

$$x = (x_1, \ldots, x_n)$$

where the component functions f_i assemble together as

$$F(x) = (f_1(x), \ldots, f_n(x)), \quad \forall\, x \in X$$

and the fixed point of F is computed using an asynchronous distributed version of the iterative method

$$x_i = f_i(x), \quad i = 1, \ldots, n$$

Witsenhausen's Intrinsic Model

To manage the multiagent competition that occurs between unconscious processes modeled by the topos of coalgebras, we build on the intrinsic model. Witsenhausen introduced an elegant model of asynchronous distributed multiagent decision making in his *intrinsic model*, which we generalized to the categorical setting previously [111]. We briefly summarize Witsenhausen's framework as defined in our Universal Decision Model framework, and explain its relevance to modeling decision making in consciousness.

[111] Sridhar Mahadevan. Universal decision models. *CoRR*, abs/2110.15431, 2021. URL https://arxiv.org/abs/2110.15431

We briefly explain our previous work on a categorial generalization of Witsenhausen's framework, which we termed the Universal Decision Model (UDM)[112]. In the UDM category $\mathcal{C}_{\text{UDM}}$, as in any category, we are given a collection of *decision objects* $\mathcal{D}$, and a set of morphisms $\mathcal{M}_{\text{UDM}}$ between UDM objects, where $f : c \to d$ is a morphism that maps from UDM object c to d. A morphism need not exist between every pair of UDM objects. In this paper, we restrict ourselves to *locally small* UDM categories, meaning that exists only a set's worth of morphisms between any pair of UDM objects.

[112] Sridhar Mahadevan. Universal decision models. *CoRR*, abs/2110.15431, 2021. URL https://arxiv.org/abs/2110.15431

Definition 63. *A Universal Decision Model (UDM) is defined as a category $\mathcal{C}_{UDM}$, where each decision object is represented as a tuple $\langle (A, (\Omega, \mathcal{B}, P), U_\alpha, \mathcal{F}_\alpha, \mathcal{I}_\alpha)_{\alpha \in A} \rangle$, where A in URL represent coalgebras, $(\Omega, \mathcal{B}, P)$ is a probability space representing the inherent stochastic state of nature due to randomness, U_α is a measurable space from which a decision $u \in U_\alpha$ is chosen by decision object α. Each element's policy in a decision object is any function $\pi_\alpha : \prod_\beta U_\beta \to U_\alpha$ that is measurable from its information field $\mathcal{I}_\alpha$, a subfield of the overall product space $(\prod_\alpha U_\alpha, \prod_\alpha \mathcal{F}_\alpha)$, to the σ-algebra $\mathcal{F}_\alpha$. The policy of decision object α can be any function $\pi_\alpha : \prod_\beta U_\beta \to U_\alpha$.*

Definition 64. *The* **information field** *of an element $\alpha \in A$ in a decision object c in UDM category $\mathcal{C}_{UDM}$ is denoted as $\mathcal{I}_\alpha \subset \mathcal{F}_A(A)$ characterizes the information available to decision object α for choosing a decision $u \in U_\alpha$.*

To ground this definition out in terms of the stochastic approximation theory of Q-learning, an information field precisely delineates what information is available to each of the parallel asynchronous distributed processors that are updating the Q-function. The information field structure yields a surprisingly rich topological space that has many important consequences for how to organize the decision makers in a complex organization into subsystems. An element α in a decision object requires information from other elements or subsystems in the network. To formalize this notion, we use product decision fields and product σ-algebras, with their canonical projections.

Definition 65. *Given a subset of nodes $B \subset A$, let $H_B = \Omega \times \prod_{\alpha \in B} U_\alpha$ be the* **product space of decisions** *of nodes in the subset B, where the* **product σ-algebra** *is $\mathcal{B} \times \prod_{\alpha \in B} \mathcal{F}_\alpha = \mathcal{F}_B(B)$. It is common to also denote the product σ-algebra by the notation $\otimes_{\alpha \in A} \mathcal{F}_\alpha$. If $C \subset B$, then the* **induced**

σ-**algebra** $\mathcal{F}_B(C)$ *is a subfield of* $\mathcal{F}_B(B)$*, which can also be viewed as the inverse image of* $\mathcal{F}_C(C)$ *under the canonical projection of* H_B *onto* H_C*.* [113]

[113] Note that for any cartesian product of sets $\prod_i X_i$, we are always able to uniquely define a projection map into any component set X_i, which is a special case of the product universal property in a category.

Universal Reinforcement Learning

Our framework for consciousness models unconscious processes by coalgebras, which collectively must compete with each other to post their data in short-term memory. In the CTM, this process is modeled by a binary tree. In our framework, this tree data structure is generalized to a functor diagram, over which we apply our recent Universal Reinforcement Learning (URL) framework to model the competition. In this section, we briefly review UR. The problem of minimization of a vector function $F : X \to X$ is generalized to finding a final coalgebra with a specified F-dynamics, where F is some functor in a symmetric monoidal category $\mathcal{C}$ with tensor product $\otimes$.

Mapping Unconscious to Conscious Memory as a Network Economy

In the final section, we explore modeling the process of multiagent decision making among the large number of unconscious processes, modeled as coalgebras, as a *network economy*. The fundamental problem is that short-term memory is a constrained resource, and we believe it is natural to explore applying the principle of network economics to this problem. Broadly, a network economy consists of a group of autonomous agents that share a network, which are divided into producer agents, transporter agents, and consumer agents. To draw the parallel to modeling consciousness, the producer agents are the unconscious processes. The transporter agents are in charge of the problem of transporting (i.e., in the brain, the neural pathways leading to the area where conscious short-term memory resides) information from the unconscious long-term memory into short-term memory. Finally, the consumer agents are locations in short-term memory that can be seen as “bidding" for different combinations of producer and transporter agents.

A Network Economic Model of Consciousness

Let us consider modeling consciousness as a collection of *producer* agents that want to post information in short-term memory from their unconscious processing (e.g., recalling some information from long-term memory), a set of *transporter* agents that manage the neural pathways from long-term memory into short-term memory whose task is to transport the information generated by the producer agents, and finally a set of *consumer* agents corresponding to locations in short-term memory that must choose products from some combination of producer and transporter agents. The broad idea here is that information filters into short-term memory from long-term memory through a competitive bidding process. The mathematics of network economics involves *variational inequalities* (VIs) [114], and we will explore an asynchronous decentralized framework for solving VIs.

[114] A. Nagurney. *Network Economics: A Variational Inequality Approach*. Kluwer Academic Press, 1999

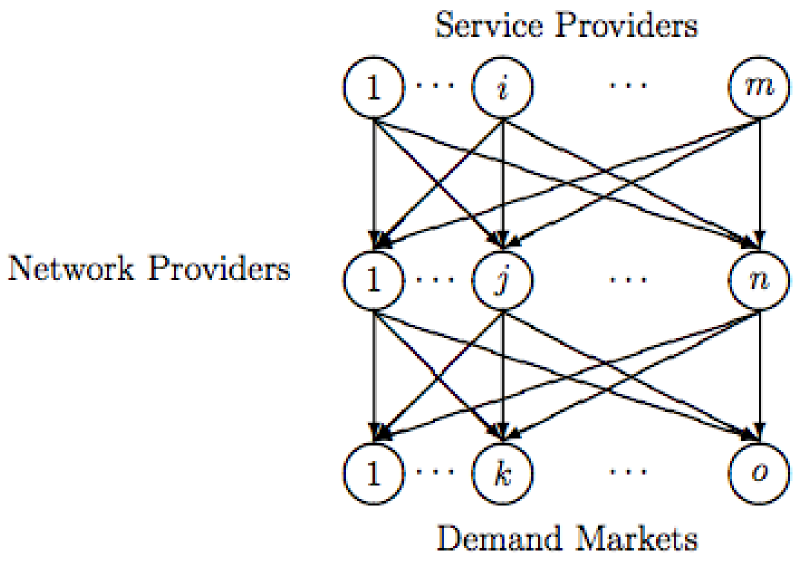


Figure 55: Modeling consciousness as a network economy. The top tier represent producer agents from long-term unconscious memory who want to post their data into short-term conscious memory, but must bid and compete with other producer agents for the privilege. The middle tier are transporter agents, charged with the task of moving the information from long-term to short-term memory along neural pathways. Finally, the bottom tier are consumer agents that correspond to locations in short-term memory, which must choose some combination of producer and transporter agents to display the desired information at their location.

In terms of the UDM framework, the set of elements in this decision object can be represented as $(A, (\Omega, \mathcal{B}, P), U_\alpha, \mathcal{F}_\alpha, \mathcal{I}_\alpha)_{\alpha \in A}$, where A is defined by the set of vertices in this graph representing the decision makers. For example, long-term memory agent i chooses its actions from the set U_i, which can be defined as $\cup_{j,k} Q_{ijk}$. $\mathcal{F}_i$ is the associated measurable space associated with U_i. $\mathcal{I}_i$ represents the information field of agent i, namely its visibility into the decisions made by other entities in the network at the current or past time steps.

Network economics is the study of a rich class of equilibrium problems that occur in the real world, from traffic management to supply chains and two-sided online marketplaces. This framework is general, and applies to electronic (e.g., finance) and material (e.g., physical) goods. Here, we are applying the framework to model consciousness. Each unconscious process has a utility function is defined in terms of the nonnegative service quantity (Q), quality (q), and price (π) delivered from producer provider i by network provider j to consumer agent k. Production costs, demand functions, delivery costs, and delivery opportunity costs are designated by f, ρ, c, and oc respectively. Unconscious process provider i attempts to maximize its utility function $U_i^1(Q, q^*, \pi^*)$ by adjusting Q_{ijk}. Likewise, network provider j attempts to maximize its utility function $U_j^2(Q^*, q, \pi)$ by adjusting q_{ijk} and π_{ijk}.

$$U_i^1(Q, q^*, \pi^*) = \sum_{j=1}^{n} \sum_{k=1}^{o} \hat{\rho}_{ijk}(Q, q^*) Q_{ijk} - \hat{f}_i(Q) - \sum_{j=1}^{n} \sum_{k=1}^{o} \pi^*_{ijk} Q_{ijk}, \quad Q_{ijk} \geq 0$$

$$U_j^2(Q^*, q, \pi) = \sum_{i=1}^{m}\sum_{k=1}^{o} \pi_{ijk} Q^*_{ijk} - \sum_{i=1}^{m}\sum_{k=1}^{o} (c_{ijk}(Q^*, q) + oc_{ijk}(\pi_{ijk})), q_{ijk}, \pi_{ijk} \geq 0$$

Game Theory and Variational Inequalities

We now give a brief overview of traditional game theory, and contrast it with the VI framework. Game theory was pioneered by von Neumann and Morgenstern [115], and later extended by Nash [116]. A finite, n-person normal form game is a tuple (N, A, U), where N is a finite set of n players indexed by i, $A = A_1 \times \cdots \times A_n$ is the joint action space formed from the set actions available to each player (A_i), and U is a tuple of the players' utility functions (or payoffs) u_i where $u_i : A \to \mathbb{R}$. The difficulty in computing the equilibrium depends on the constraints placed on the game. For instance, two-player, zero-sum games, ensure that player interests are diametrically opposed and can thus be formulated as linear programs (LPs) by the minmax theorem and solved in polynomial time [117].

[115] J. von Neumann and O. Morgenstern. *Theory of games and economic behavior*. Princeton University Press, 1947

[116] John Nash. Non-cooperative games. *Annals of Mathematics*, 54(2):286–295, 1951. URL https://doi.org/10.2307/1969529

[117] Noam Nisan, Tim Roughgarden, Eva Tardos, and Vijay Vazirani. *Algorithmic Game Theory*. Cambridge University Press, Cambridge; New York, 2007. ISBN 9780521872829 0521872820

By contrast, in two-player, *general*-sum games, any increase in one player's utility does not necessarily result in a decrease in the other player's utility so a convenient LP formulation is not possible. Finding Nash equilibria in two-player, general-sum games is thought to be time exponential in the size of the game in the worst case. It has been shown that every game has at least one Nash equilibrium which delegates the problem to the class PPAD (polynomial parity argument, directed version) originally designated by Papadimitriou. Although this game type cannot be converted to an LP, it can be formulated as a linear complimentarily problem (LCP). In crude terms, the LCP can be formed by introducing an additional constraint called a complementarity condition to the combination of constraints that would appear in each agent's LP had it only been a zero-sum game. Unlike the LP, the LCP is only composed of constraints making it a pure constraint satisfaction problem (CSP). The most popular game theoretic algorithm for solving these LCPs is the Lemke-Howson algorithm. This algorithm performs a series of pivot operations that swap out player strategies until all constraints are satisfied. An alternate approach is to employ heuristics as in the case of the support-enumeration method (SEM) which repeatedly tests whether a Nash equilibrium exists given a pair of actions, or *support profile*. The heuristic used is to favor testing smaller, more balanced support profiles in order to prune larger regions of the action space.

Finally, we encounter n-player, general-sum games, in which the complementarity problem previously defined is now nonlinear (NCP). One common approach is to approximate the solution of the NCP as solving a *sequence* of LCPs (SLCP). This method is typically fast, however, it is not globally convergent. Another technique is to solve an optimization problem in which the global minima equate to the Nash equilibria of the original problem. The drawback is that there are local minima that do not correspond to Nash equilibria making global convergence difficult.

Some games exhibit a characteristic of *payoff independence* where a single player's payoff is dependent on only a subset of the other players in the game. In this case, the reward table indexed by the joint action-space of all players is overly costly prompting a move from the normal form representation of the game to the more compact representation offered by graphical games. This can often reduce the space of the representation from exponential to polynomial in the number of players. When the graph is a tree, a commonly used method, NashProp, computes an ϵ-Nash equilibrium with a back and forth sweep over the graph from the leaves to the root.

Summary and Further Reading

We described a novel theory of consciousness as a *functor* (CF) that receives and transmits contents from unconscious memory into conscious memory. CF models the ensemble of unconscious processes as a topos category of coalgebras. As every topos has an internal language defined by a Mitchell-B'enabou language with a Kripke-Joyal semantics, CF is based on an internal "language of thought" using the Multi-modal Universal Mitchell-B'enabou Language Embedding (MUMBLE). We modeled the transmission of information from conscious short-term working memory to long-term unconscious memory using our recently proposed Universal Reinforcement Learning (URL) framework. To model the transmission of information from unconscious long-term memory into short-term memory, we propose a network economic model, where "producer" agents correspond to unconscious processes, "transporter" agents correspond to neural pathways from long-term to short-term memory, and "consumer agents" correspond to short-term memory locations that use a competitive bidding process to manage the competition between unconscious long-term memory processes. Both URL and the network economic model of consciousness build on a formal theoretical framework for asynchronous parallel distributed computation without the need for synchronization by a global clock.

There are a wealth of books on consciousness in many fields, ranging from cognitive science to psychology and philosophy. I recommend Baars' book, as it provides the most influential current model of consciousness as the stream of data from conscious short-term memory to unconscious long-term memory, which influenced this chapter. [118]

[118] Bernard Baars. *In the theater of consciousness: The workspace of the mind.* Oxford Univ. Press, New York, NY, 1997. URL https://psycnet.apa.org/doi/10.1093/acprof:oso/9780195102659.001.1

Bibliography

Bernard Baars. *In the theater of consciousness: The workspace of the mind*. Oxford Univ. Press, New York, NY, 1997. URL `https://psycnet.apa.org/doi/10.1093/acprof:oso/9780195102659.001.1`.

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

Dimitri P. Bertsekas and John N. Tsitsiklis. *Parallel and Distributed Computation: Numerical Methods*. Athena Scientific, 1997. ISBN 1886529019.

Patrick Billingsley. *Probability and Measure*. Wiley, 3rd edition, 1995.

Raoul Bott and Loring W. Tu. *Differential Forms in Algebraic Topology*. Springer, 1982.

Tai-Danae Bradley, John Terilla, and Yiannis Vlassopoulos. An enriched category theory of language: from syntax to semantics, 2021. Arxiv.

Philippe Brouillard, Sébastien Lachapelle, Alexandre Lacoste, Simon Lacoste-Julien, and Alexandre Drouin. Differentiable causal discovery from interventional data, 2020. URL `https://arxiv.org/abs/2007.01754`.

Gunnar E. Carlsson and Facundo Mémoli. Classifying clustering schemes. *Found. Comput. Math.*, 13(2):221–252, 2013. DOI: 10.1007/s10208-012-9141-9. URL `https://doi.org/10.1007/s10208-012-9141-9`.

Sneha Chaudhari, Varun Mithal, Gungor Polatkan, and Rohan Ramanath. An attentive survey of attention models, 2021. URL `https://arxiv.org/abs/1904.02874`.

David Maxwell Chickering. Optimal structure identification with greedy search. *Journal of machine learning research*, 3(Nov):507–554, 2002.

Herbert Edelsbrunner, David Letscher, and Afra Zomorodian. Topological persistence and simplification. *Discrete & Computational Geometry*, 28 (4):511–533, 2002.

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

Brendan Fong, David I. Spivak, and Rémy 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.

Tobias Fritz. A synthetic approach to markov kernels, conditional independence and theorems on sufficient statistics. *Advances in Mathematics*, 370: 107239, August 2020. ISSN 0001-8708. DOI: 10.1016/j.aim.2020.107239. URL http://dx.doi.org/10.1016/j.aim.2020.107239.

Robert Ghrist. *Elementary Applied Topology*. Createspace, 2014.

Mustafa Hajij, Lennart Bastian, Sarah Osentoski, Hardik Kabaria, John L. Davenport, Sheik Dawood, Balaji Cherukuri, Joseph G. Kocheemoolayil, Nastaran Shahmansouri, Adrian Lew, Theodore Papamarkou, and Tolga Birdal. Copresheaf topological neural networks: A generalized deep learning framework, 2025. URL https://arxiv.org/abs/2505.21251.

Guido W. Imbens and Donald B. Rubin. *Causal Inference for Statistics, Social, and Biomedical Sciences: An Introduction*. Cambridge University Press, USA, 2015. ISBN 0521885884.

Amin Jaber, Murat Kocaoglu, Karthikeyan Shanmugam, and Elias Bareinboim. Causal discovery from soft interventions with unknown targets: Characterization and learning. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 9551–9561. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/6cd9313ed34ef58bad3fdd504355e72c-Paper.pdf.

Bart 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.

Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani. *An Introduction to Statistical Learning – with Applications in R*, volume 103 of *Springer Texts in Statistics*. Springer, New York, 2013. ISBN 978-1-4614-7137-0. DOI: 10.1007/DOI.

Dominik Janzing, David Balduzzi, Moritz Grosse-Wentrup, and Bernhard Schölkopf. Quantifying causal influences. *Annals of Statistics*, 41(5): 2324–2358, 2013.

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

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

Urvashi Khandelwal, Omer Levy, Dan Jurafsky, Luke Zettlemoyer, and Mike Lewis. Generalization through memorization: Nearest neighbor language models, 2020. URL https://arxiv.org/abs/1911.00172.

Jeonghoon Kim, Byeongchan Lee, Cheonbok Park, Yeontaek Oh, Beomjun Kim, Taehwan Yoo, Seongjin Shin, Dongyoon Han, Jinwoo Shin, and Kang Min Yoo. Peri-ln: Revisiting normalization layer in the transformer architecture, 2025. URL https://arxiv.org/abs/2502.02732.

Jon Kleinberg. An impossibility theorem for clustering. In S. Becker, S. Thrun, and K. Obermayer, editors, *Advances in Neural Information Processing Systems*, volume 15. MIT Press, 2002.

Saunders Mac Lane and Ieke Moerdijk. *Sheaves in Geometry and Logic a First Introduction to Topos Theory*. Springer New York, New York, NY, 1992. ISBN 9781461209270 1461209277. URL http://link.springer.com/book/10.1007/978-1-4612-0927-0.

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

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

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

Sridhar Mahadevan. GAIA: Categorical foundations of generative AI, 2024. URL https://arxiv.org/abs/2402.18732.

Sridhar Mahadevan. Consciousness as a functor, 2025a. URL https://arxiv.org/abs/2508.17561.

Sridhar Mahadevan. Decentralized causal discovery using judo calculus, 2025b. URL https://arxiv.org/abs/2510.23942.

Sridhar Mahadevan. Intuitionistic j-do-calculus in topos causal models, 2025c. URL https://arxiv.org/abs/2510.17944.

Sridhar Mahadevan. Large causal models from large language models, 2025d. URL https://arxiv.org/abs/2512.07796.

Sridhar Mahadevan. Topos theory for generative AI and LLMs, 2025e. URL https://arxiv.org/abs/2508.08293.

Sridhar Mahadevan. Universal causal inference in a topos. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025f. URL https://openreview.net/forum?id=TOhpnECT10.

Sridhar Mahadevan. Universal reinforcement learning in coalgebras: Asynchronous stochastic computation via conduction, 2025g. URL https://arxiv.org/abs/2508.15128.

Sridhar Mahadevan. Rethinking AI: From functions to functors. In *Proceedings of the Fortieth AAAI Conference on Artificial Intelligence, January 20-27, 2026, Singapore*, 2026. URL https://people.cs.umass.edu/~mahadeva/papers/AAAI_2026_SM_Talk_Rethinking_AI.pdf.

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

A. Nagurney. *Network Economics: A Variational Inequality Approach*. Kluwer Academic Press, 1999.

John Nash. Non-cooperative games. *Annals of Mathematics*, 54(2):286–295, 1951. URL https://doi.org/10.2307/1969529.

Noam Nisan, Tim Roughgarden, Eva Tardos, and Vijay Vazirani. *Algorithmic Game Theory*. Cambridge University Press, Cambridge; New York, 2007. ISBN 9780521872829 0521872820.

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

Ben Poole, Subhaneil Lahiri, Maithra Raghu, Jascha Sohl-Dickstein, and Surya Ganguli. Exponential expressivity in deep neural networks through transient chaos. *Advances in Neural Information Processing Systems*, 29, 2016. URL https://arxiv.org/abs/1606.05340.

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

Karen Sachs, Omar Perez, Dana Pe'er, Douglas A. Lauffenburger, and Garry P. Nolan. Causal protein-signaling networks derived from multiparameter single-cell data. *Science*, 308(5721):523–529, 2005. DOI: 10.1126/science.1105809.

Samuel S. Schoenholz, Justin Gilmer, Surya Ganguli, and Jascha Sohl-Dickstein. Deep information propagation. *International Conference on Learning Representations*, 2017. URL https://arxiv.org/abs/1611.01232.

Ana 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.

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

Leslie G. Valiant. A theory of the learnable. *Commun. ACM*, 27(11):1134–1142, 1984. DOI: 10.1145/1968.1972. URL https://doi.org/10.1145/1968.1972.

Ruben van Belle. *Kan Extensions in Probability Theory*. PhD thesis, University of Edinburgh, 2024.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman 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.

J. von Neumann and O. Morgenstern. *Theory of games and economic behavior*. Princeton University Press, 1947.

H. S. Witsenhausen. The intrinsic model for discrete stochastic control: Some open problems. In A. Bensoussan and J. L. Lions, editors, *Control Theory, Numerical Methods and Computer Systems Modelling*, pages 322–335, Berlin, Heidelberg, 1975. Springer Berlin Heidelberg.

Tong Xiao and Jingbo Zhu. Introduction to Transformers: an NLP perspective, 2023. URL https://arxiv.org/abs/2311.17633.

Ruibin Xiong, Yaru Yang, Di He, Kai Zhang, Shuxin Zheng, Hao Zheng, Chen Xing, Liangchen Liu, and Liwei Wang. On layer normalization in the transformer architecture. *arXiv preprint arXiv:2002.04745*, 2020.

Greg Yang. Scaling limits of wide neural networks with weight sharing. *Advances in Neural Information Processing Systems*, 32, 2019. URL https://arxiv.org/abs/1902.04760.

Chulhee Yun, Srinadh Bhojanapalli, Ankit Singh Rawat, Sashank J. Reddi, and Sanjiv 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.

Alessio Zanga and Fabio Stella. A survey on causal discovery: Theory and practice, 2023. URL https://arxiv.org/abs/2305.10032.

Xun Zheng, Bryon Aragam, Pradeep Ravikumar, and Eric P Xing. Dags with no tears: Continuous optimization for structure learning. In *Advances in Neural Information Processing Systems*, 2018.