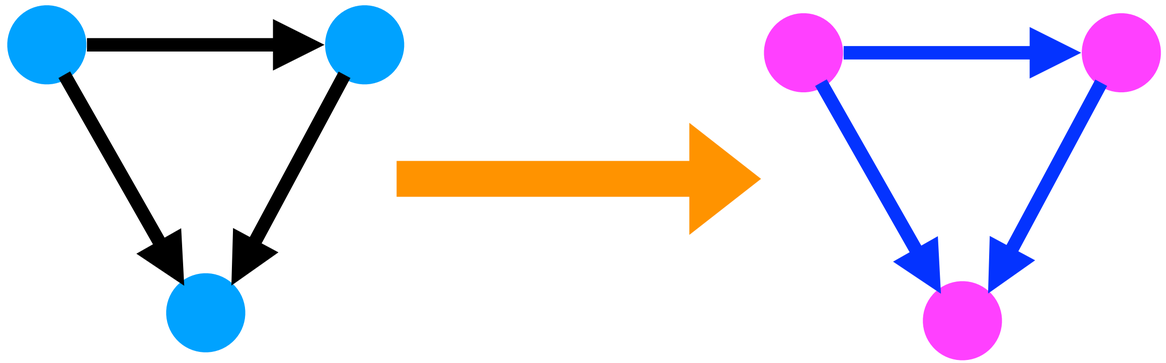
Thinking with Functors: Category Theory for A(G)I

AAAI 2025 Tutorial TH18, February 26, 8:30am -12:30 pm

Sridhar Mahadevan
Adobe and University of Massachusetts, Amherst



Goals and Outline of Tutorial

- To teach you abstraction and compositional thinking:
  - How to “think with categories and functors”
  - How to analyze A(G)I systems using functors
- To cover theory and practice
  - Illustrate applications of categories and functors to A(G)I
  - Introduce new directions for A(G)I research

Nothing is more practical than a good theory.

~ Ludwig Boltzmann

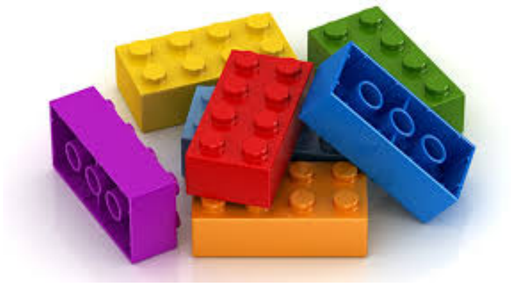
AZ QUOTES

“A theory is not like an airline or bus timetable. We are not interested simply in the accuracy of its predictions. A theory also serves as a base for thinking. It helps us to understand what is going on by enabling us to organize our thoughts.”

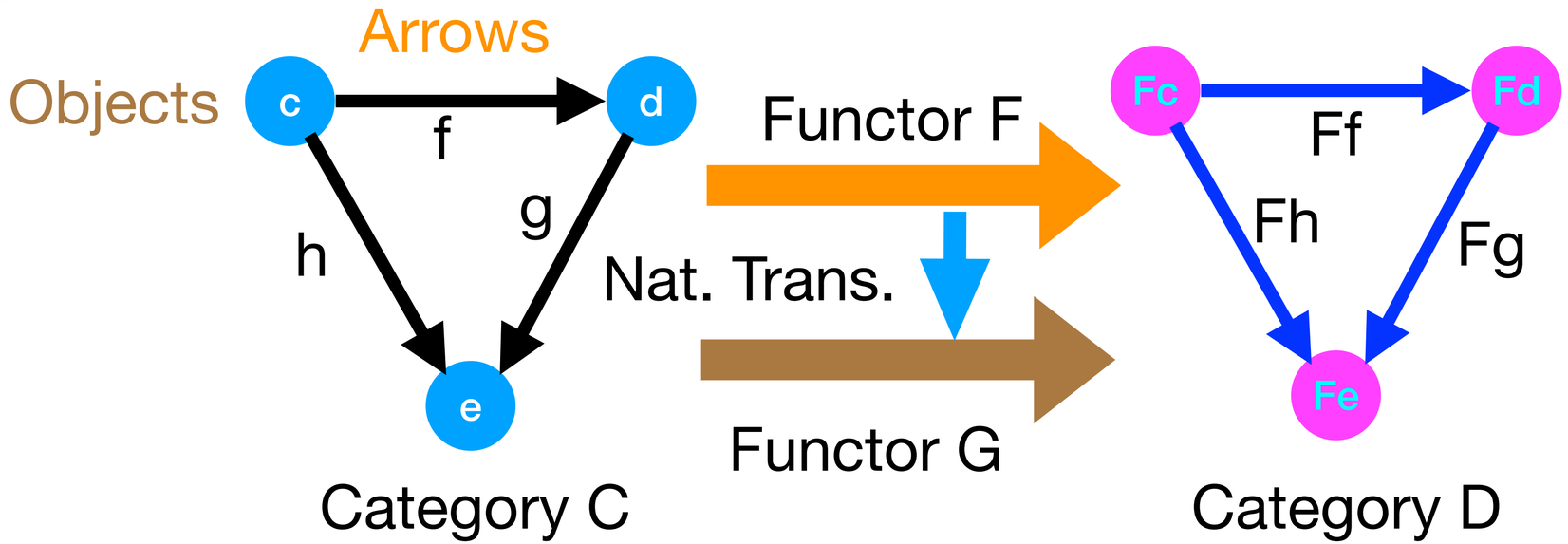
—Ronald Coase

Introduction to Category Theory

Abstraction and Compositionality



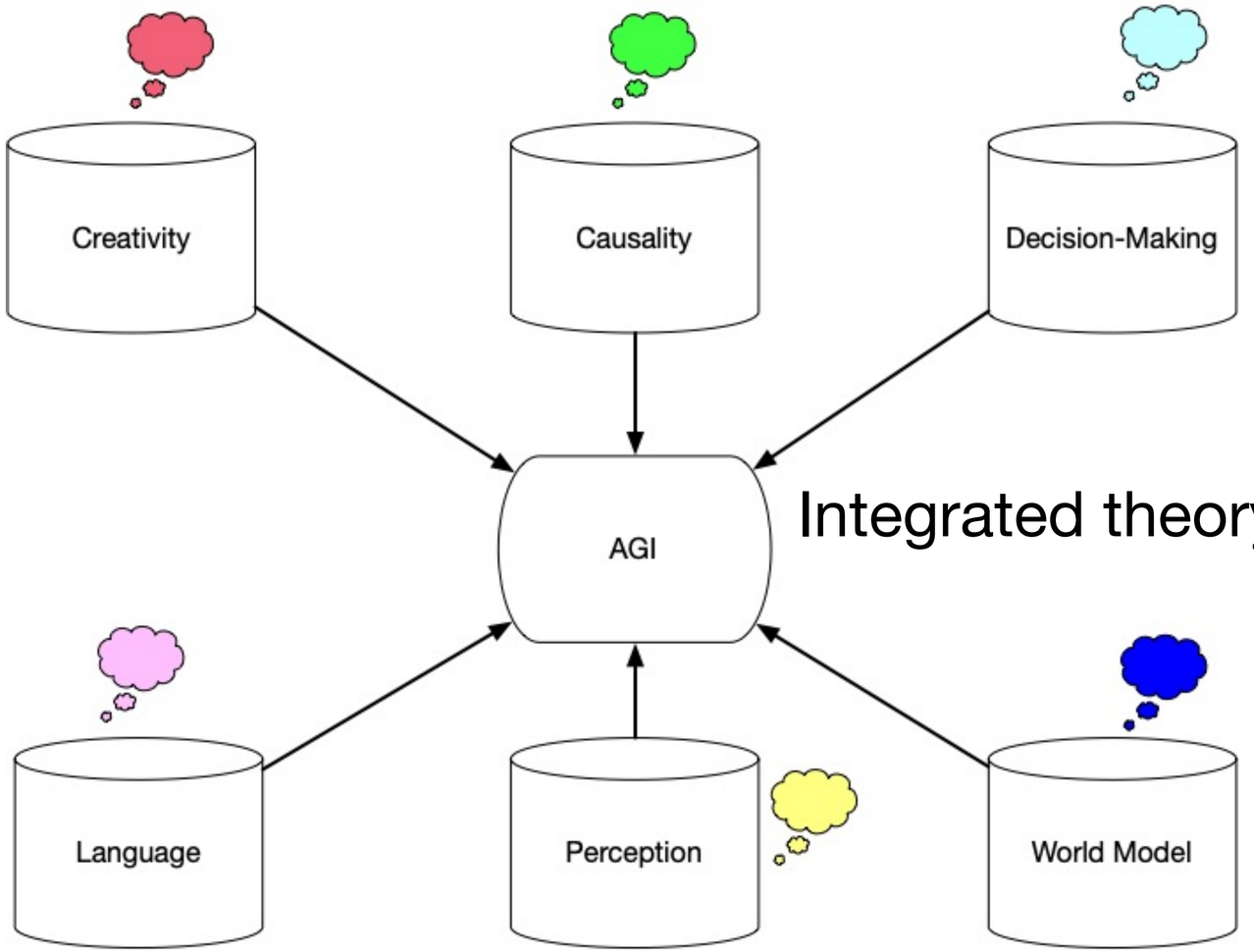
The Building Blocks



Objects: Causal models or Transformer models

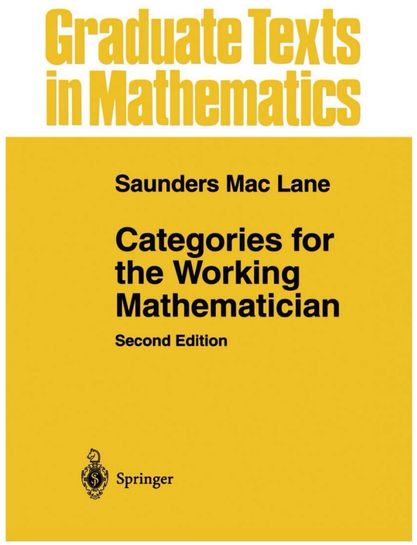
Arrows: Causal interventions or compositions of Transformer blocks

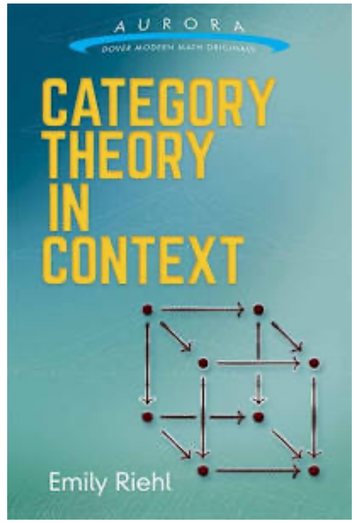
A Compositional Theory of AGI

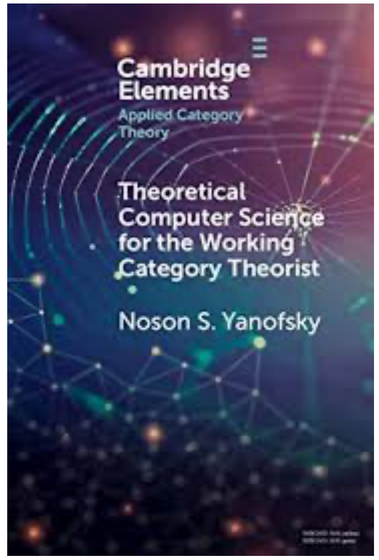


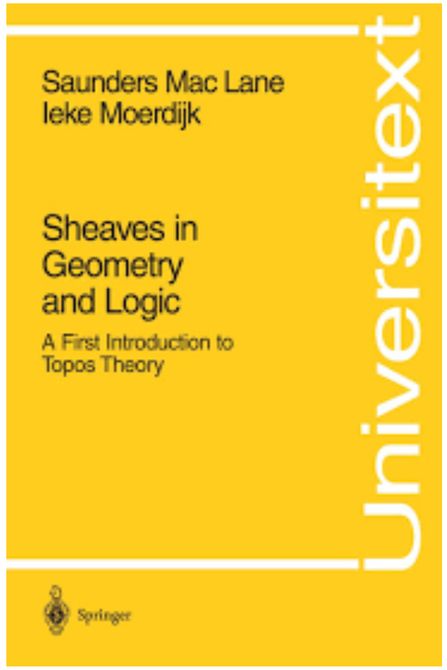
Internal languages

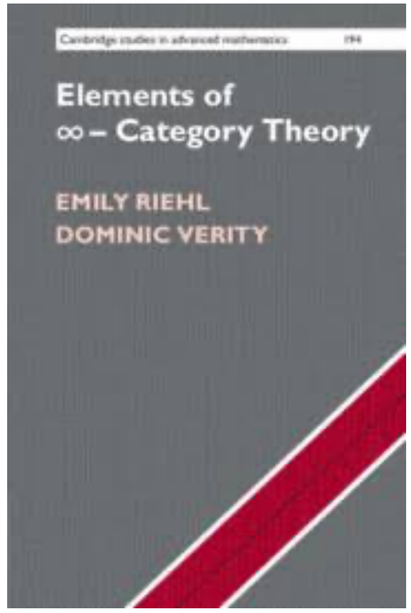
Integrated theory of mind

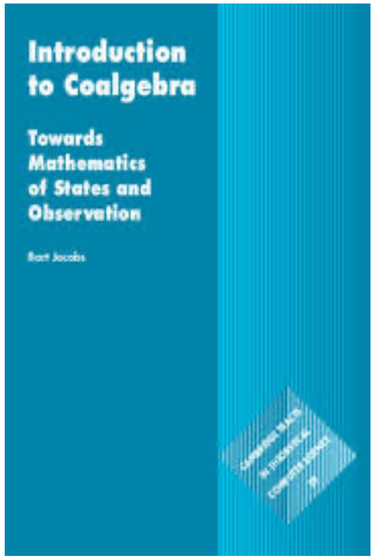




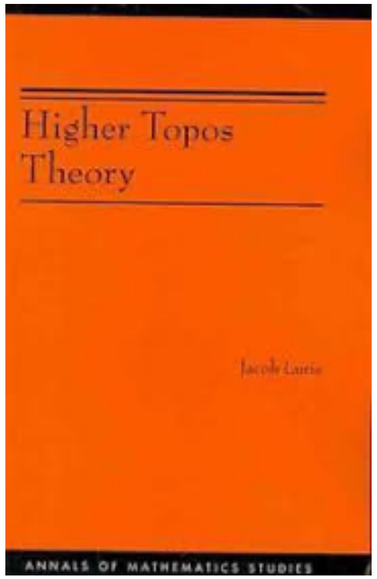








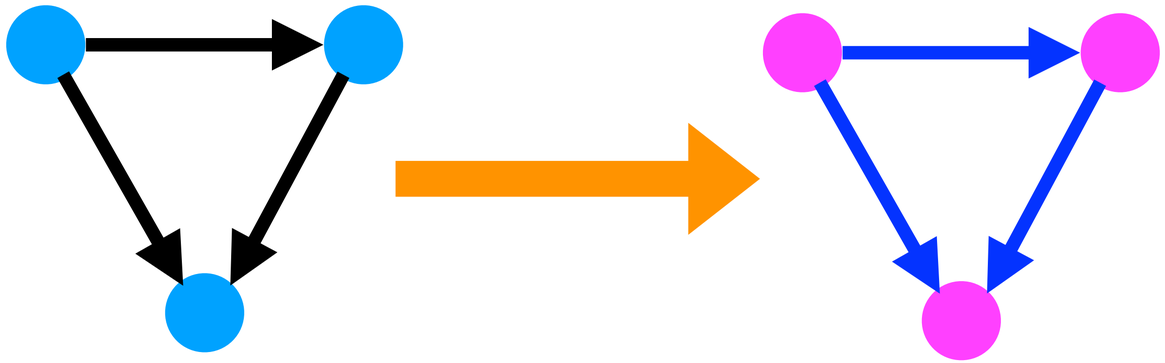
Basic → Advanced



What is Category Theory?

- It is the most transformative unification in mathematics since set theory
- It merges many different subfields of mathematics
  - Geometry to Logic: theory of sheaves on a topos
  - Algebra to Probability: monads
  - Induction vs. coinduction: universal coalgebras as dynamical systems

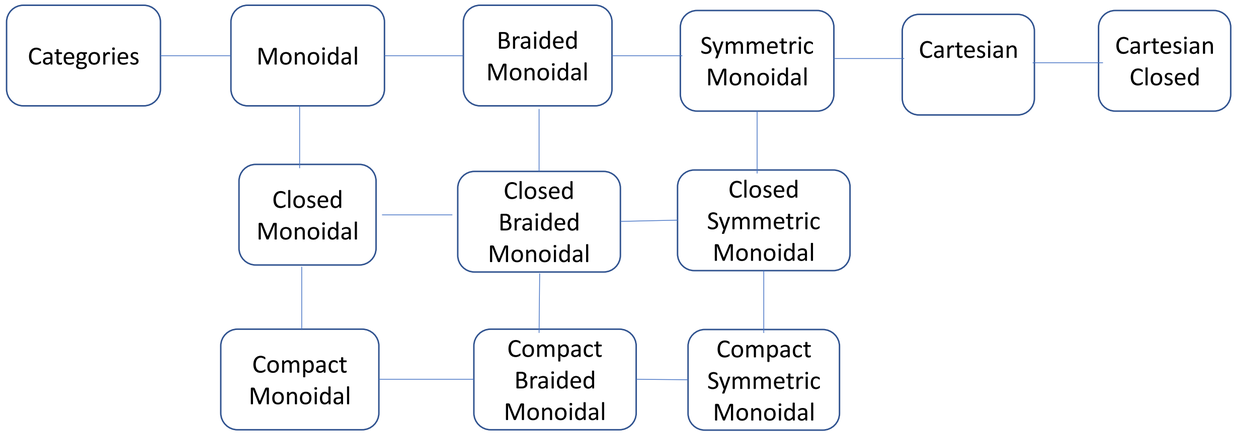
What are Categories?



- A category C is defined by
  - A collection of objects (x, y, z, …)
  - A collection of arrows C(x,y)
  - Compositionality: x —> y —> z compose in general
- Remarkable property: The category **Cat** of all categories is a category!
  - What are the objects of **Cat**?
  - What are the arrows of **Cat**?

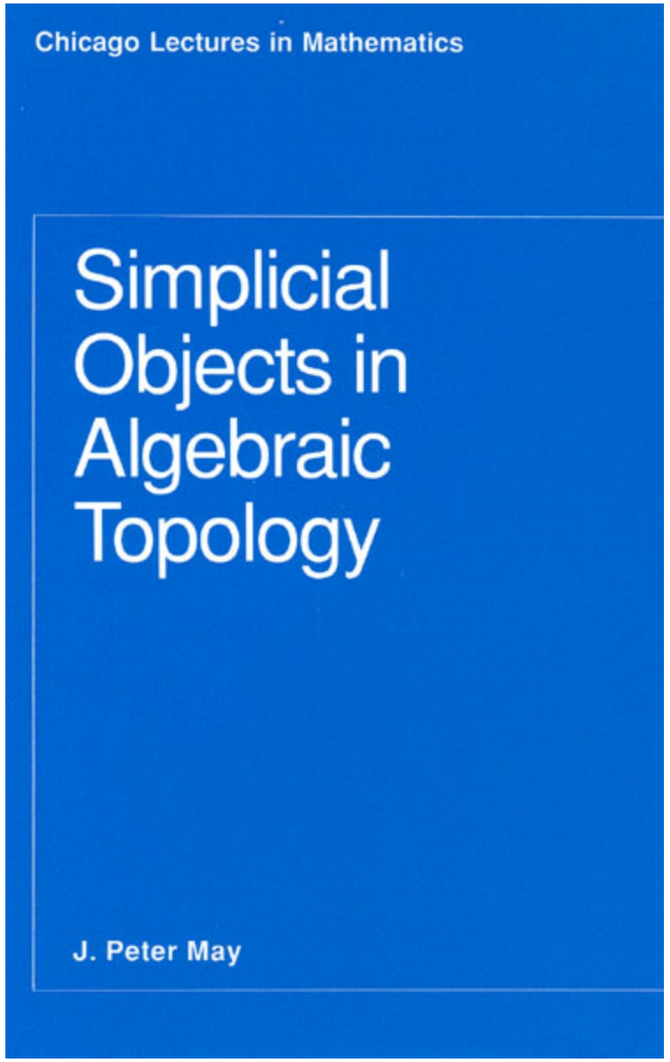
There are a myriad ways to construct categories!

Monoidal Categories

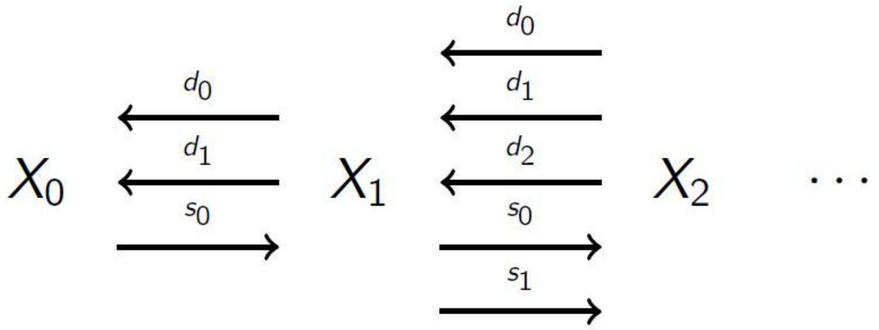


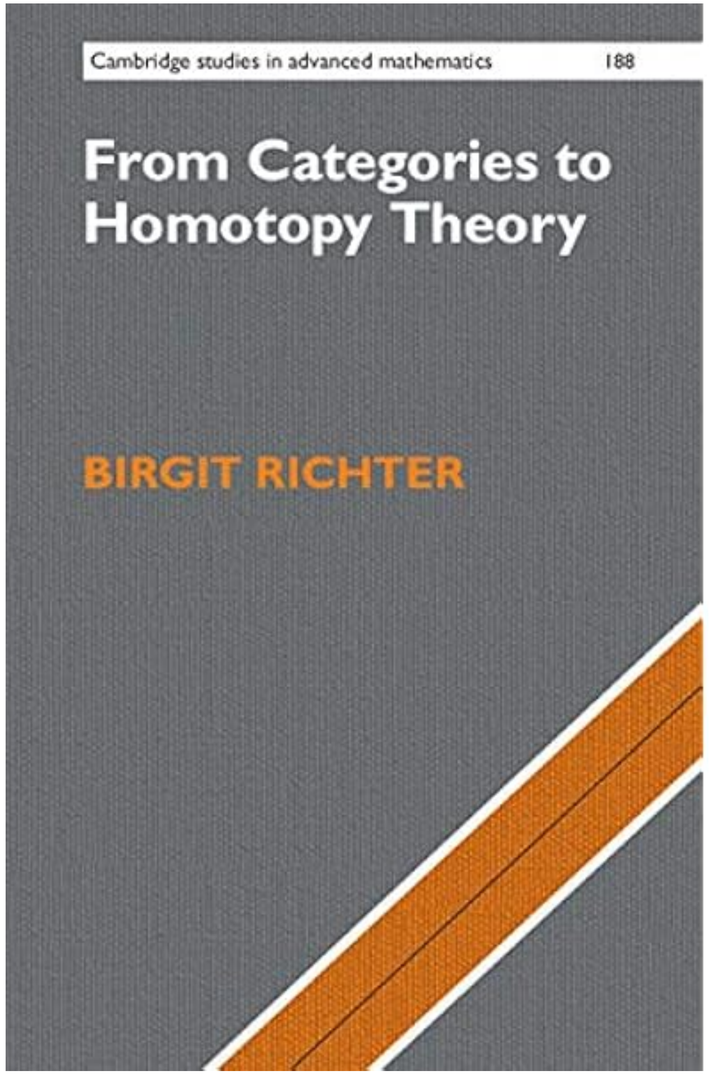
- Equipped with a tensor product:
  - $\otimes : C \times C \to C$
  - Identity element 1: $1 \otimes c \simeq c \simeq c \otimes 1$
- Unit interval [0,1]: closed symmetric monoidal preorder
- $\mathscr{V}-$enriched monoidal category: $a, b \in C \Rightarrow C(a, b) \in \mathscr{V}$

Simplicial Objects



Combinatorial model of spaces





Main Concepts of Category Theory

Functors

Yoneda Lemma

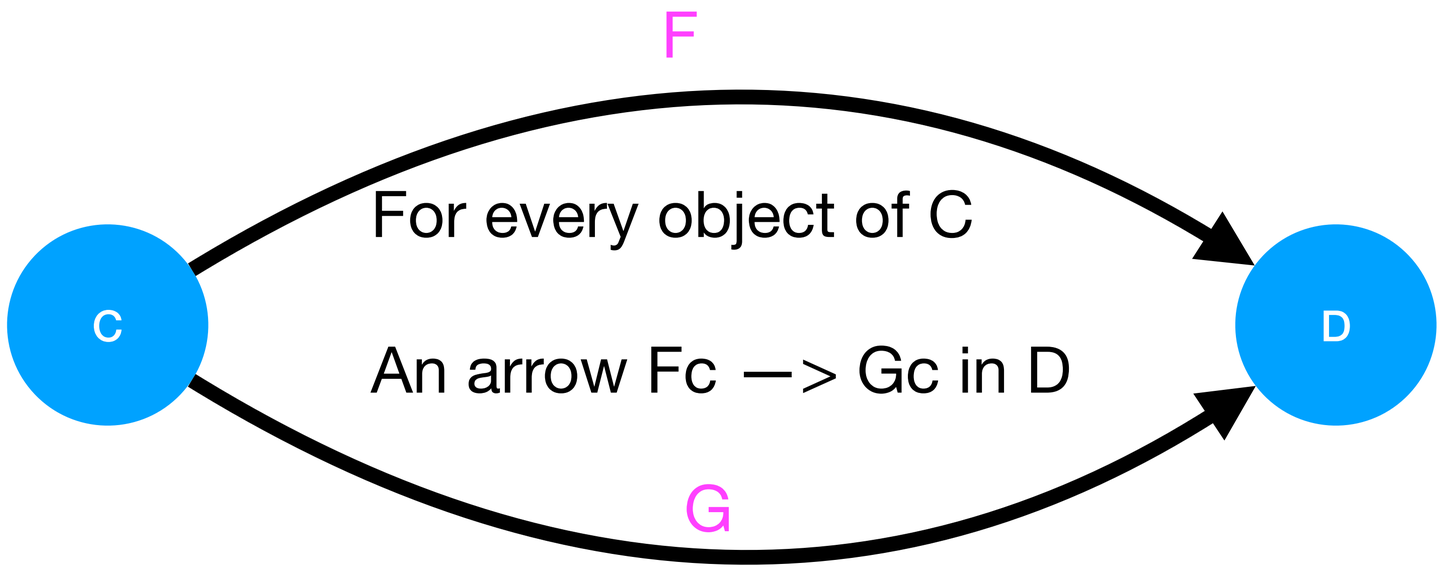
Adjoint Functors

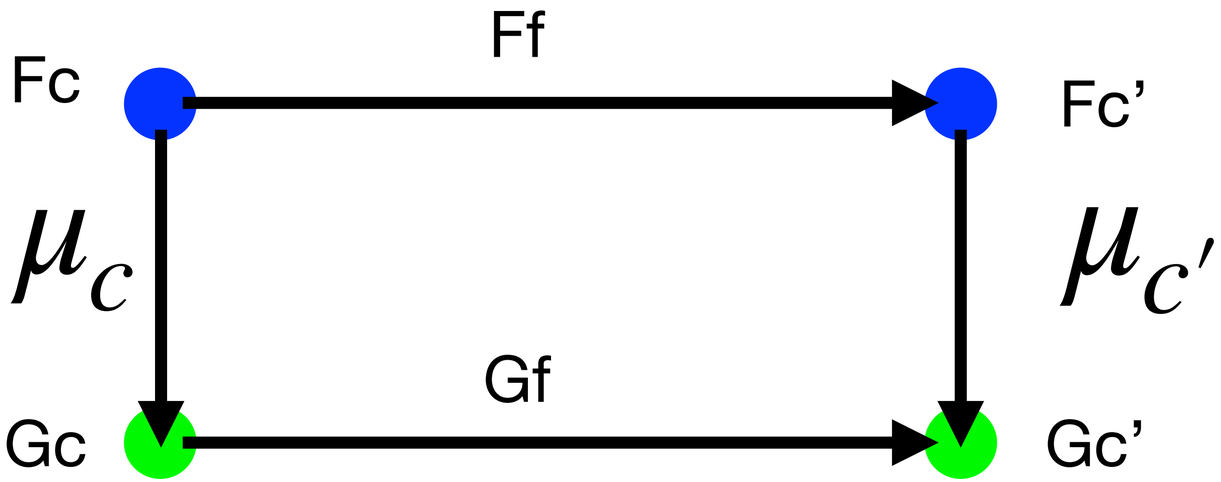
Limits Colimits

Monads

Kan Extensions

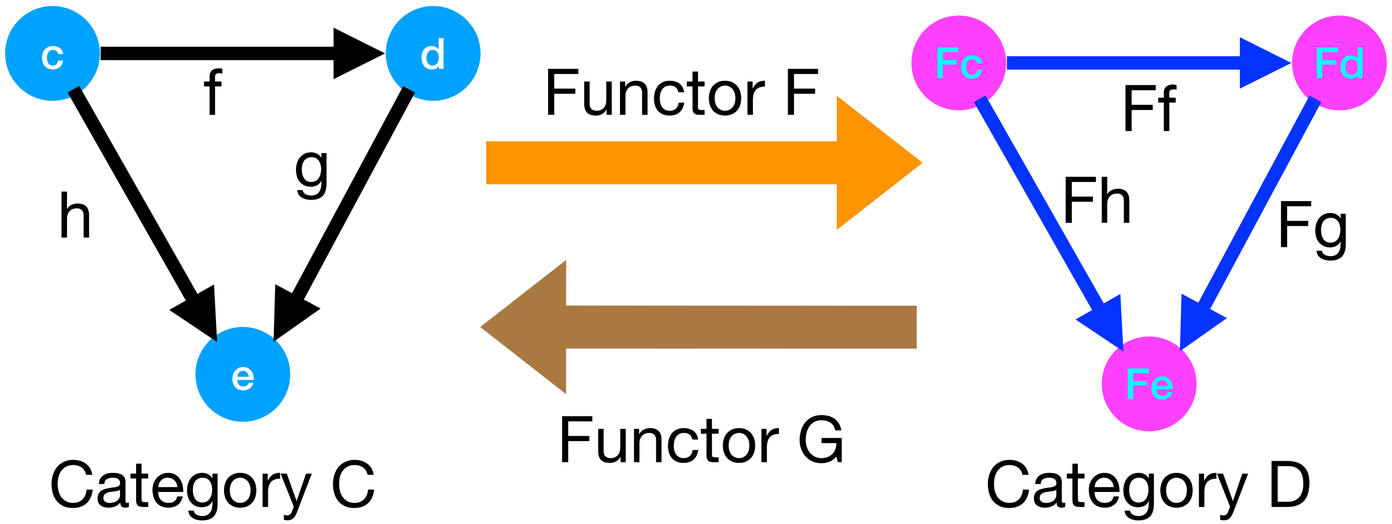
Natural Transformations between Functors





Adjoint Functors

C(c,Gd) ~ D(Fc, d)



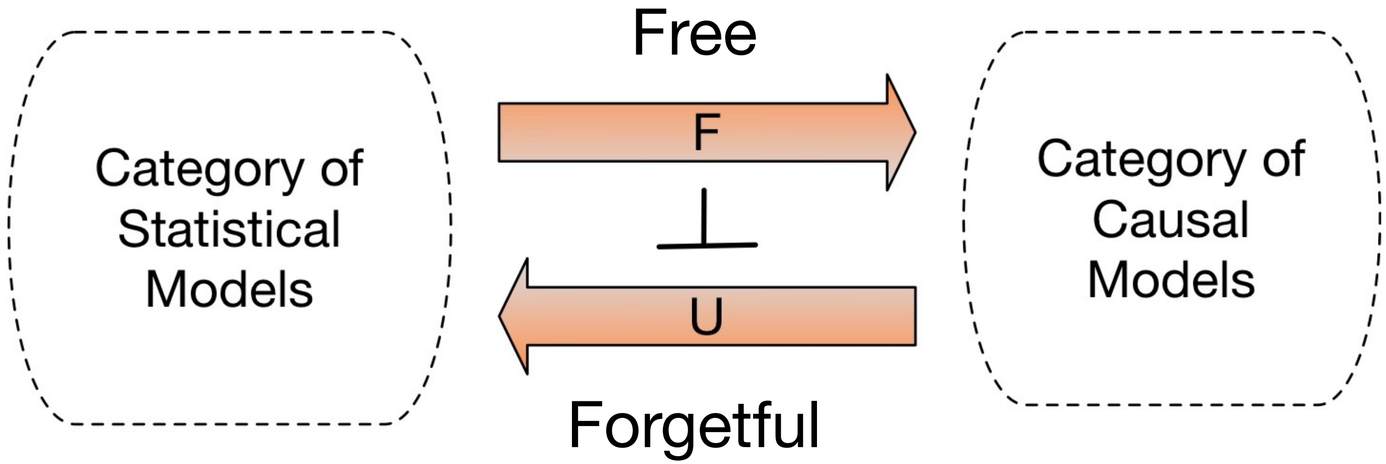
“Free” and “Forgetful” functors:

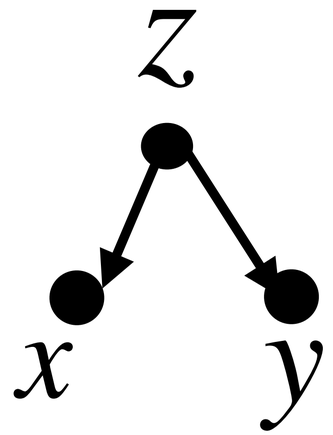
Sets —> Abelian Groups

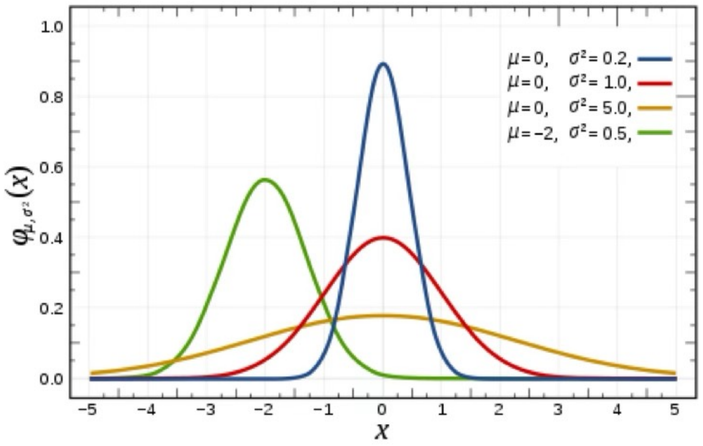
Categories —> Graphs

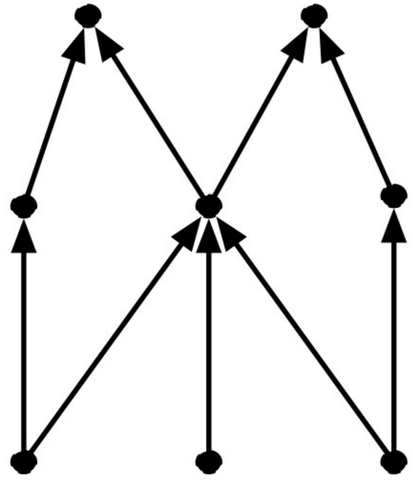
Adjoint Functors

$$x \perp y \mid z$$



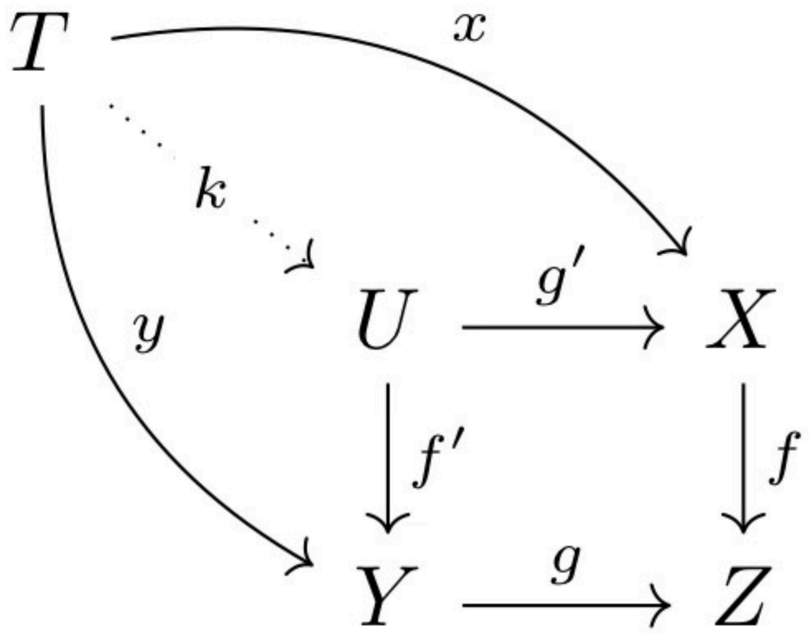






Universal Property: Limits/Colimits

- The Cartesian product of two sets is a set
  - Can we define it without looking ``inside" a set?
  - Find its universal property
- Limits: Equalizer, meet, supremum, …
- Colimits: Coequalizer, join, infimum, …



Pullback

If X, Y, Z are sets, U = {x | fx = gx}

If g is monic, then pullback is $f^{-1}(Z)$

“Categories”

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

Yoneda Embedding

“metric spaces,

vector spaces,…”

Birds and Frogs

Freeman Dyson

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. I happen to be a frog, but many of my best friends are birds. The main theme of my talk tonight is this. Mathematics needs both birds and frogs. Mathematics is rich and beautiful because birds give it broad visions and frogs give it intricate details. Mathematics is both great art and important science, because it combines generality of concepts with depth of structures. It is stupid to claim that birds are better than frogs because they see farther, or that frogs are better than birds because they see deeper. The world of mathematics is both broad and deep, and we need birds and frogs working together to explore it.

This talk is called the Einstein lecture, and I am grateful to the American Mathematical Society for inviting me to do honor to Albert Einstein. Einstein was not a mathematician, but a physicist who had mixed feelings about mathematics. On the one hand, he had enormous respect for the power of mathematics to describe the workings of nature, and he had an instinct for mathematical beauty which led him onto the right track to find nature's laws. On the other hand, he had no interest in pure mathematics, and he had no technical skill as a mathematician. In his later years he hired younger colleagues with the title of assistants to do mathematical calculations for him. His way of thinking was physical rather than mathematical. He was supreme among physicists as a bird who saw further than others. I will not talk about Einstein since I have nothing new to say.

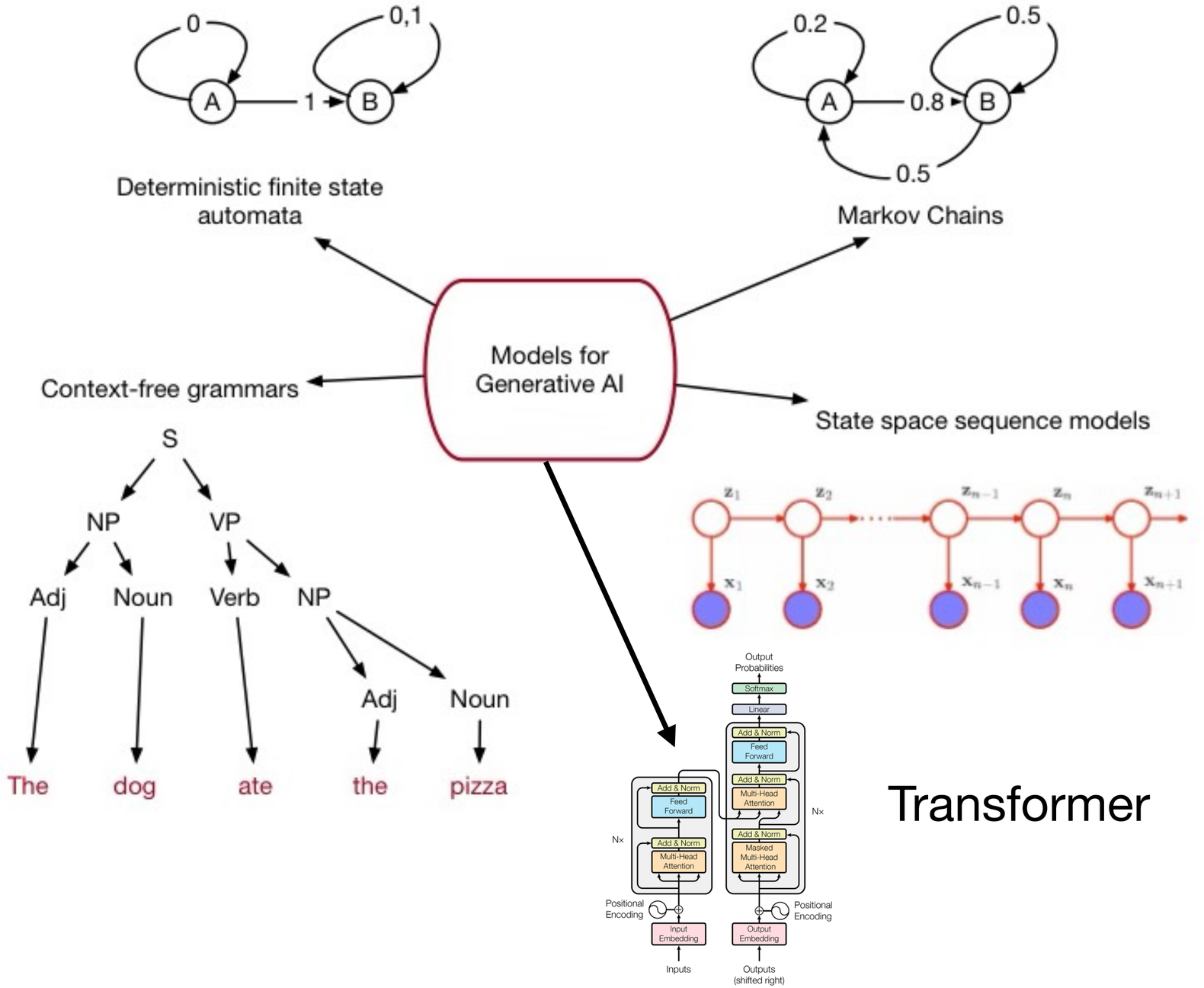
Francis Bacon and René Descartes

At the beginning of the seventeenth century, two great philosophers, Francis Bacon in England and René Descartes in France, proclaimed the birth of modern science. Descartes was a bird, and Bacon was a frog. Each of them described his vision of the future. Their visions were very different. Bacon said, “All depends on keeping the eye steadily fixed on the facts of nature.” Descartes said, “I think, therefore I am.” According to Bacon, scientists should travel over the earth collecting facts, until the accumulated facts reveal how Nature works. The scientists will then induce from the facts the laws that Nature obeys. According to Descartes, scientists should stay at home and deduce the laws of Nature by pure thought. In order to deduce the laws correctly, the scientists will need only the rules of logic and knowledge of the existence of God. For four hundred years since Bacon and Descartes led the way, science has raced ahead by following both paths simultaneously. Neither Baconian empiricism nor Cartesian dogmatism has the power to elucidate Nature's secrets by itself, but both together have been amazingly successful. For four hundred years English scientists have tended to be Baconian and French scientists Cartesian. Faraday and Darwin and Rutherford were Baconians; Pascal and Laplace and Poincaré were Cartesians. Science was greatly enriched by the cross-fertilization of the two contrasting cultures. Both cultures were always at work in both countries. Newton was at heart a Cartesian, using

Freeman Dyson is an emeritus professor in the School of Natural Sciences, Institute for Advanced Study, Princeton, NJ. His email address is dyson@ias.edu.

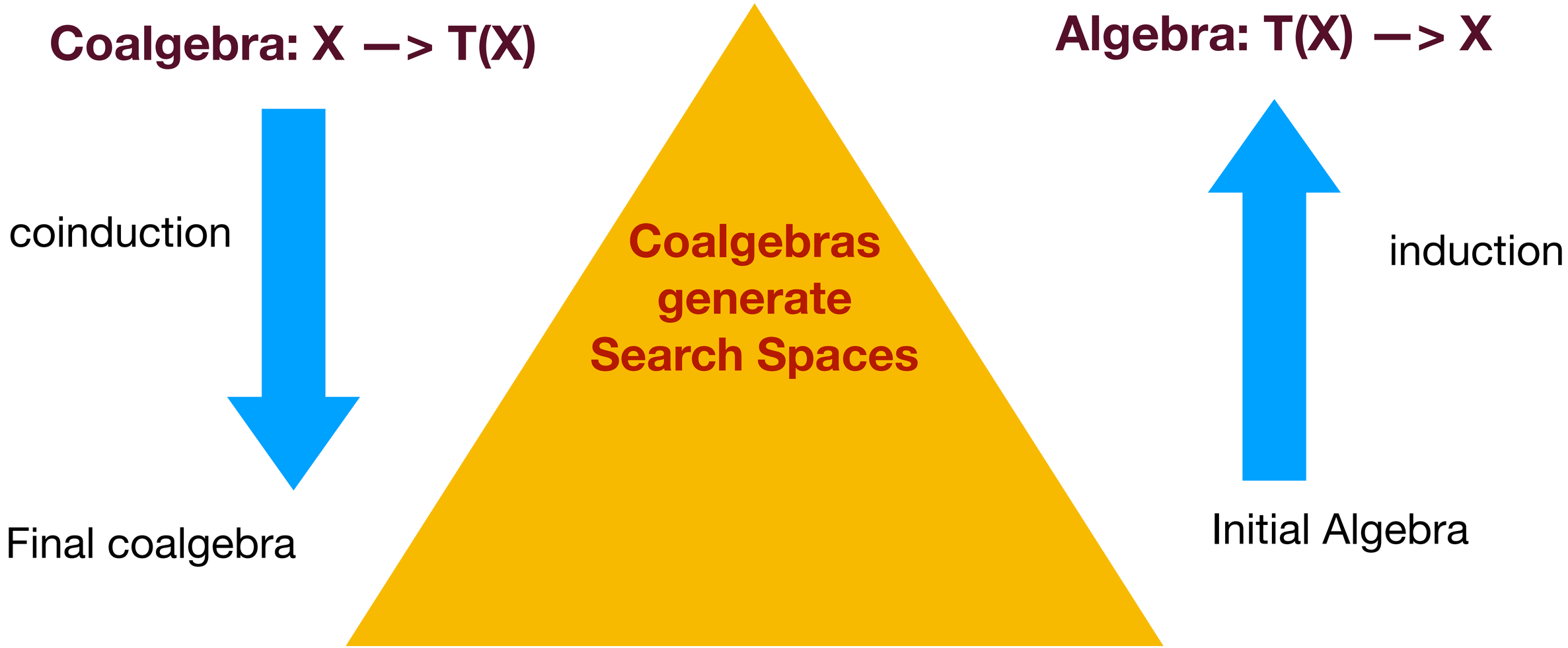
This article is a written version of his AMS Einstein Lecture, which was to have been given in October 2008 but which unfortunately had to be canceled.

Categorical Generative Models



Generative models as

Universal coalgebras


Coalgebra: X —> T(X)
coinduction
Final coalgebra
Coalgebras
generate
Search Spaces
Algebra: T(X) —> X
induction
Initial Algebra

The PowerSet Functor

- Consider the coalgebra: X —> PowerSet(X)
  - Example: X = {0,1}, PowerSet = {{}, {0}, {1}, {0,1}}
- Why is PowerSet a functor?
  - Objects: Sets, like X
  - Arrows: functions on sets f:X —> Y
- How does PowerSet act on the arrows?
  - On arrows: Powerset(f): PowerSet(X) —> PowerSet(Y)

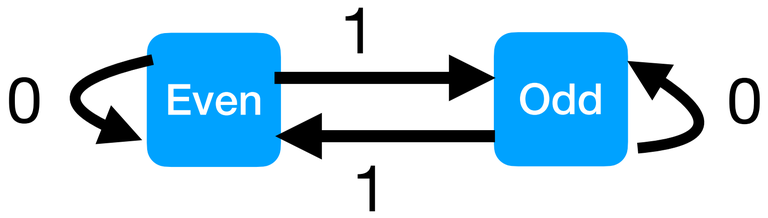
Labeled Transition Systems as Coalgebras

- **Any automata (deterministic or stochastic) is a coalgebra**
  - **Set of states S**
  - **Transition relation** $\rightarrow_S \subseteq S \times A \times S$
  - **Here,** $s \rightarrow^a t$ **is the same as** $(s, a, t) \in \rightarrow_S$
  - **Coalgebra of LTS defined by powerset functor L**
    - $\alpha_S : S \rightarrow L(S), s \mapsto \{(a, s') \mid s \rightarrow^a s'\}$

Category theory gives us a new way to define states!

(X, X —> T(X)): Coalgebra

$$(Q, \alpha : Q \mapsto (Q^A \times B))$$



$$(Q, A, B, \delta : Q \times A \to Q, \beta : Q \to B)$$

Hard attention Transformers cannot compute parity!

[Hahn, ACL, 2020]

ELSEVIER

Theoretical Computer Science 249 (2000) 3–80

Theoretical Computer Science

www.elsevier.com/locate/tcs

Fundamental Study

Universal coalgebra: a theory of systems

J.J.M.M. Rutten

CWI, P.O. Box 94079, 1090 GB Amsterdam, Netherlands

Communicated by M.W. Mislove

Abstract

In the semantics of programming, finite data types such as finite lists, have traditionally been modelled by initial algebras. Later final *coalgebras* were used in order to deal with *infinite* data types. Coalgebras, which are the dual of algebras, turned out to be suited, moreover, as models for certain types of automata and more generally, for (transition and dynamical) *systems*. An important property of initial algebras is that they satisfy the familiar principle of induction. Such a principle was missing for coalgebras until the work of Aczel (Non-Well-Founded sets, CSLI Leethre Notes, Vol. 14, center for the study of Languages and information, Stanford, 1988) on a theory of non-wellfounded sets, in which he introduced a proof principle nowadays called *coinduction*. It was formulated in terms of *bisimulation*, a notion originally stemming from the world of concurrent programming languages. Using the notion of *coalgebra homomorphism*, the definition of bisimulation on coalgebras can be shown to be formally dual to that of congruence on algebras. Thus, the three basic notions of universal algebra: algebra, homomorphism of algebras, and congruence, turn out to correspond to coalgebra, homomorphism of coalgebras, and bisimulation, respectively. In this paper, the latter are taken as the basic ingredients of a theory called *universal coalgebra*. Some standard results from universal algebra are reformulated (using the aforementioned correspondence) and proved for a large class of coalgebras, leading to a series of results on, e.g., the lattices of subcoalgebras and bisimulations, simple coalgebras and coinduction, and a covariety theorem for coalgebras similar to Birkhoff's variety theorem.

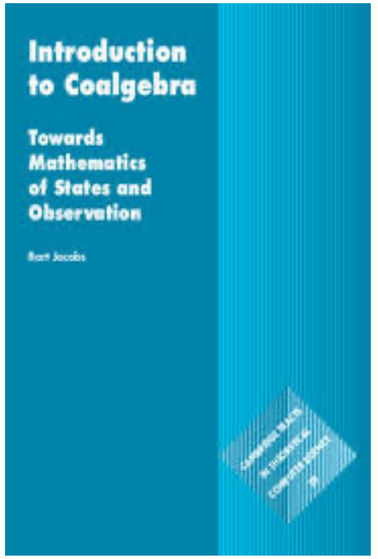
MSC: 68Q10; 68Q55

PACS: D.3; F.1; F.3

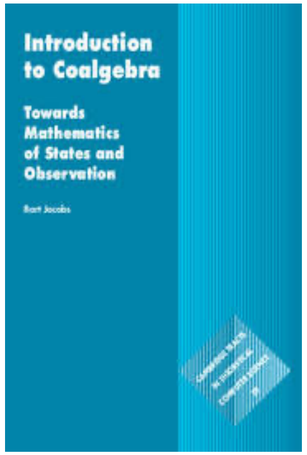
Keywords: Coalgebra; Algebra; Dynamical system; Transition system; Bisimulation; Universal coalgebra; Universal algebra; Congruence; Homomorphism; Induction; Coinduction; Variety; Covariety

E-mail address: janr@cwi.nl (J.J.M.M. Rutten).

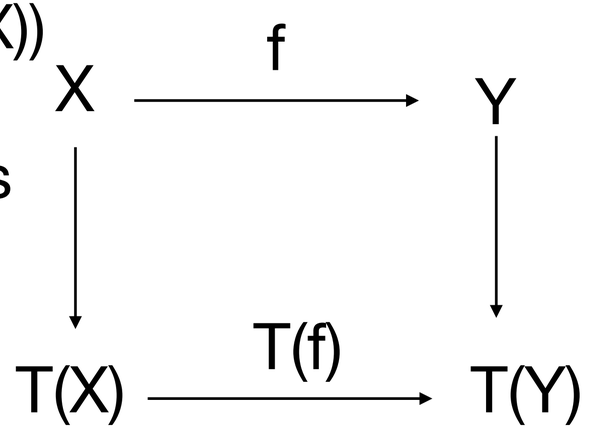
0304-3975/00/$ - see front matter
PII: S0304-3975(00)00056-6



Category of Coalgebras



- Category of Coalgebras:
  - Objects: Coalgebras (X, a: X —> T(X))
  - Arrows: Coalgebra homomorphisms



- Initial and Final Coalgebras
  - Initial object: unique morphism into other objects
  - Final object: unique morphism from other objects
- Generalizes the concept of fixed points!

Categorical Deep Learning

Deep Learning as a Functor

Backprop as Functor: A compositional perspective on supervised learning

Brendan Fong David Spivak Rémy Tuyéras

Department of Mathematics, Massachusetts Institute of Technology

Computer Science and Artificial Intelligence Lab, Massachusetts Institute of Technology

***Abstract*—A supervised learning algorithm searches over a set of functions $A \to B$ parametrised by a space P to find the best approximation to some ideal function $f\colon A \to B$. It does this by taking examples $(a, f(a)) \in A \times B$, and updating the parameter according to some rule. We define a category where these update rules may be composed, and show that gradient descent—with respect to a fixed step size and an error function satisfying a certain property—defines a monoidal functor from a category of parametrised functions to this category of update rules. A key contribution is the notion of request function. This provides a structural perspective on backpropagation, giving a broad generalisation of neural networks and linking it with structures from bidirectional programming and open games.**

Consider a supervised learning algorithm. The goal of a supervised learning algorithm is to find a suitable approximation to a function $f\colon A \to B$. To do so, the supervisor provides a list of pairs $(a, b) \in A \times B$, each of which is supposed to approximate the values taken by f, i.e. $b \approx f(a)$. The supervisor also defines a space of functions over which the learning algorithm will search. This is formalised by choosing a set P and a function $I\colon P \times A \to B$. We denote the function at parameter $p \in P$ as $I(p, -)\colon A \to B$. Then, given a pair $(a, b) \in A \times B$, the learning algorithm takes a current hypothetical approximation of f, say given by $I(p, -)$,

GAIA: Generative AI Architecture

GAIA: Categorical Foundations of Generative AI*

A Preprint

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

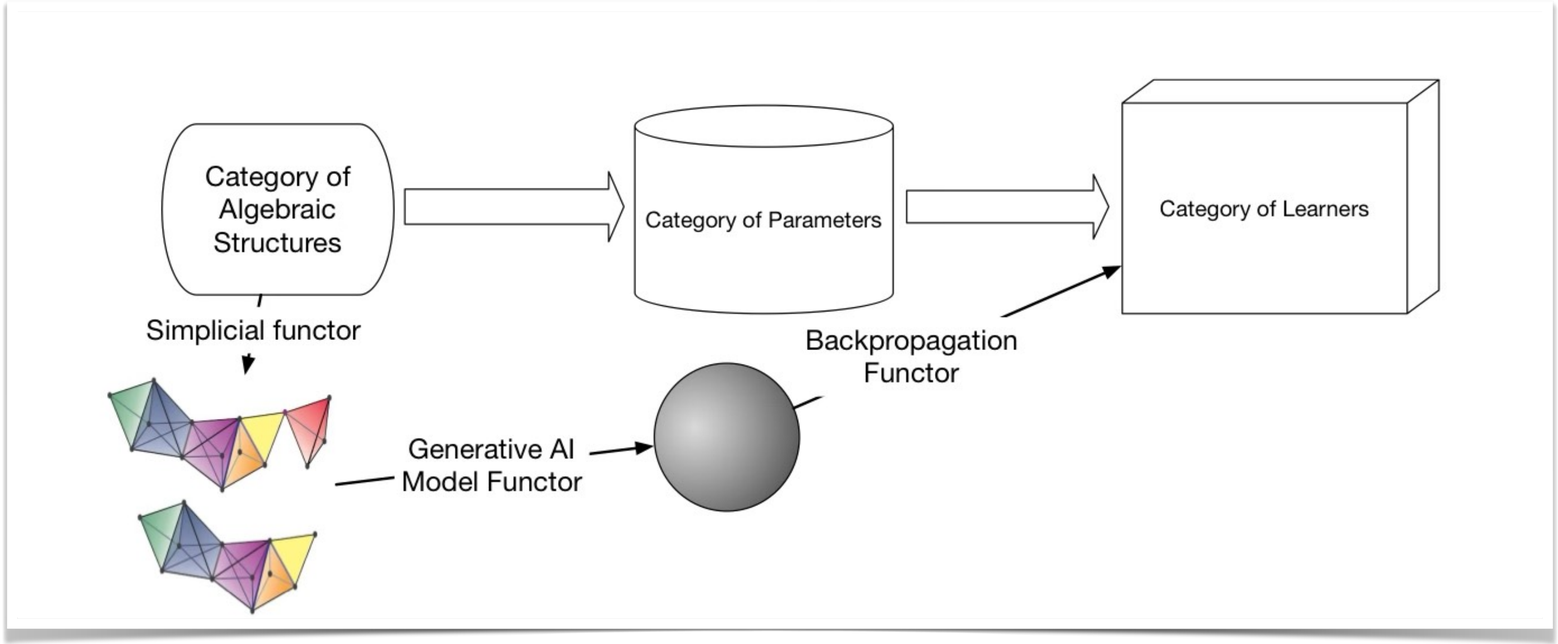
March 1, 2024

Abstract

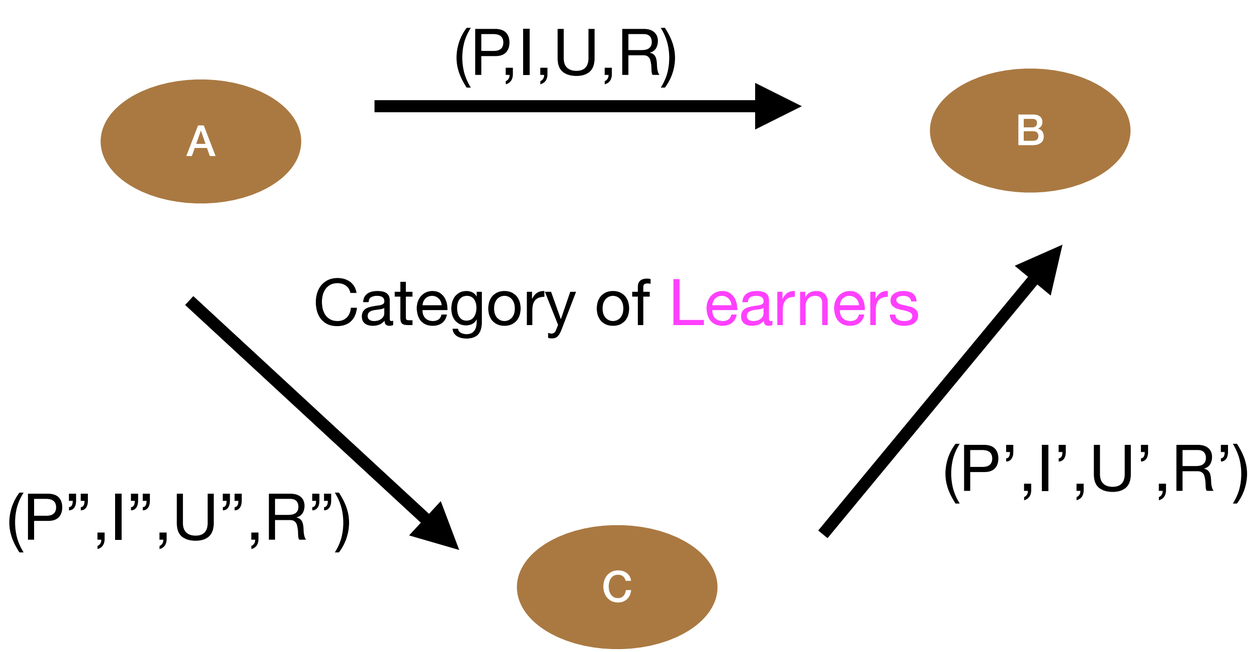
In this paper, we explore the categorical foundations of generative AI. Specifically, we investigate a Generative AI Architecture (GAIA) that lies beyond backpropagation, the longstanding algorithmic workhorse of deep learning. Backpropagation is at its core a compositional framework for (un)supervised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

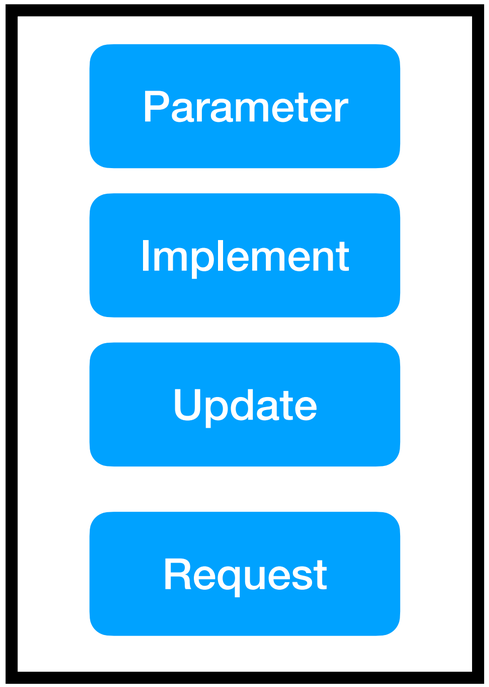
At the top level, GAIA uses the simplicial category of ordinal numbers with objects defined as $[n], n \geqslant 0$ and arrows defined as weakly order-preserving mappings $f : [n] \to [m]$, where $f(i) \leqslant f(j), i \leqslant j$. This top-level structure can be viewed as a combinatorial "factory"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

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



Compositional Deep Learning

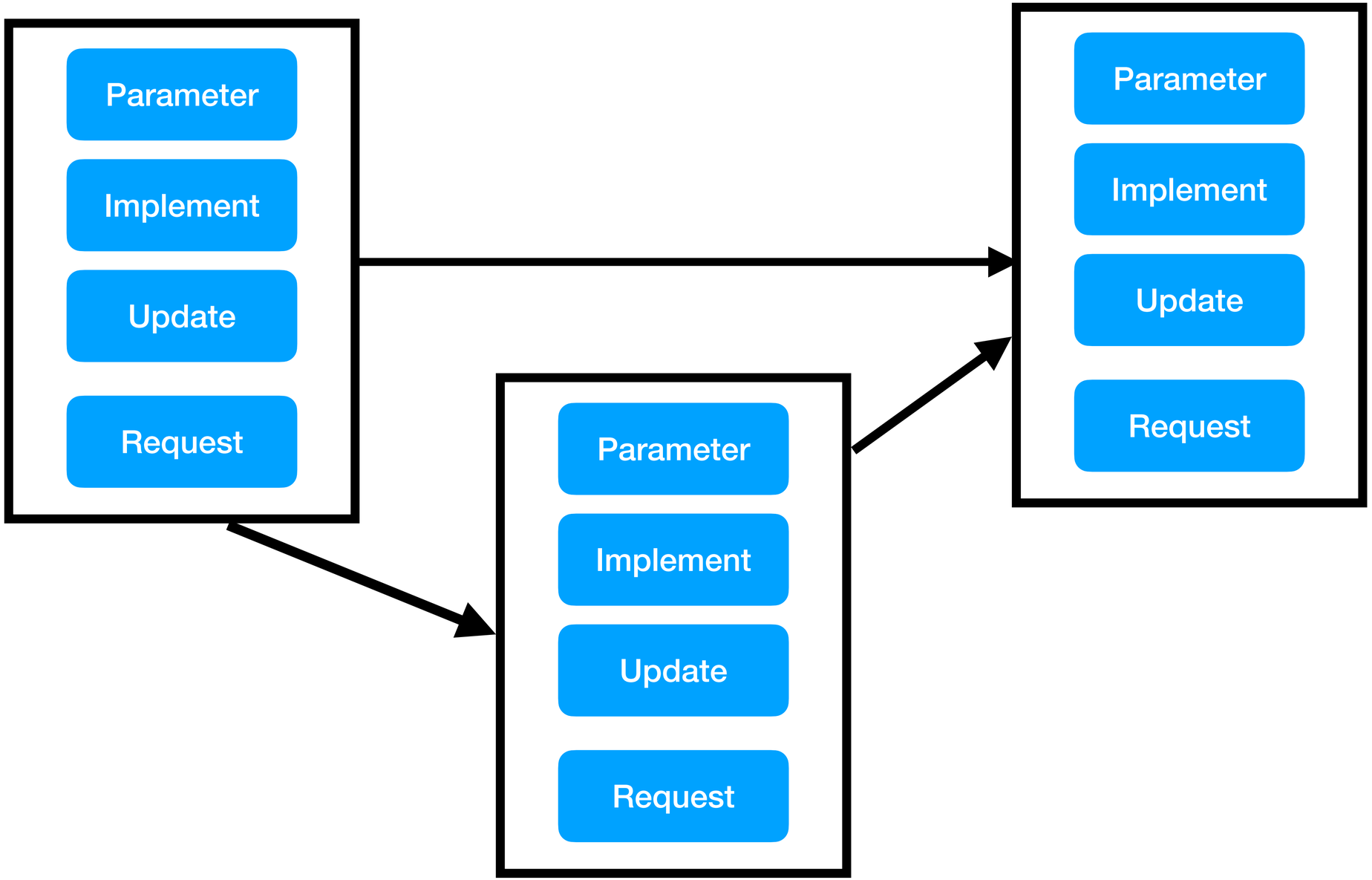




Arrows are Learners!

[Fong, Spivak, Tuyeras: “Backprop as Functor”, 2019]

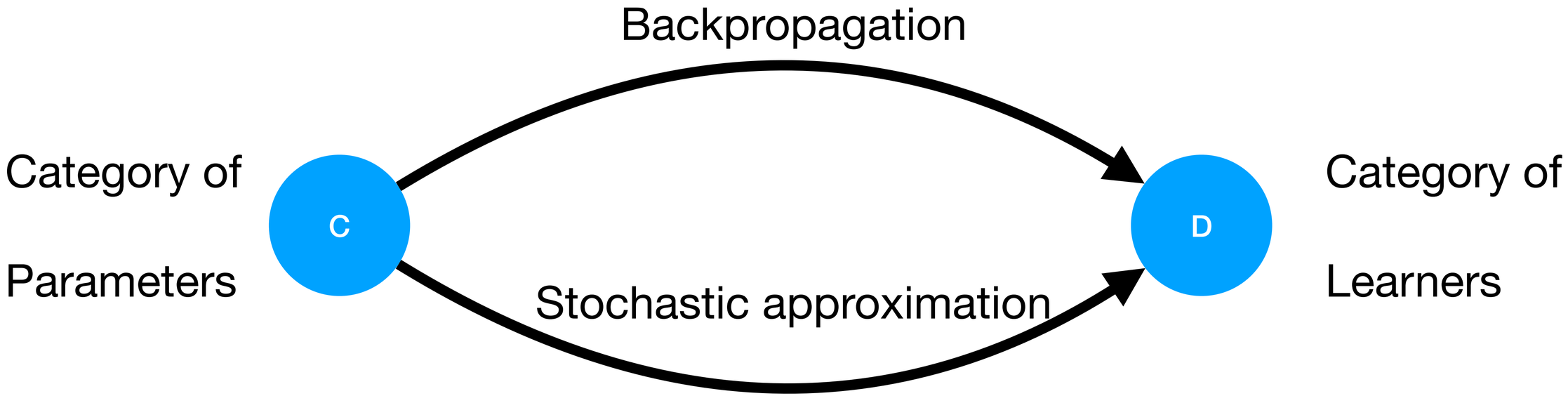
How to compose Learners?

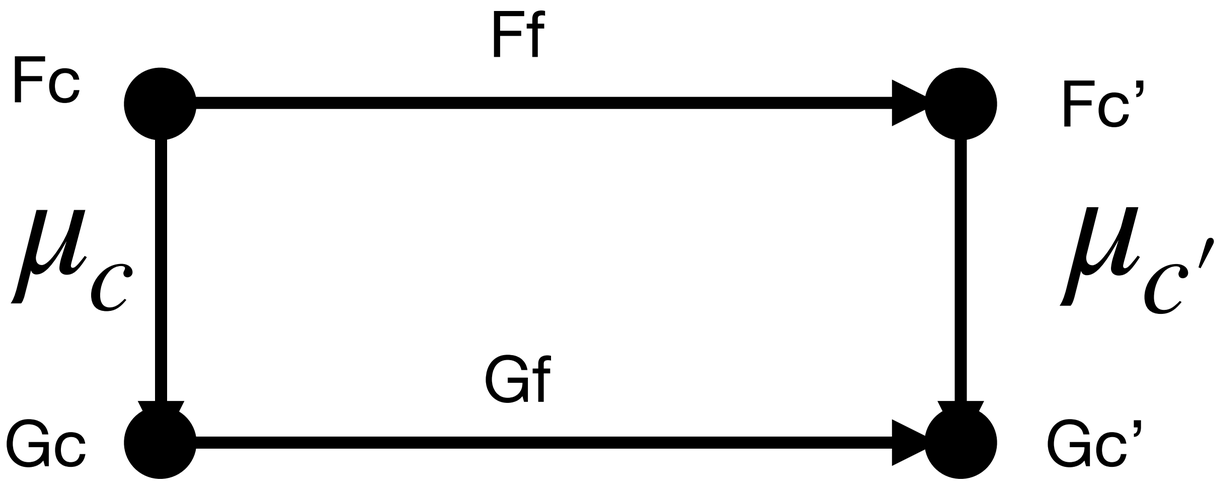


Backpropagation as a coalgebra

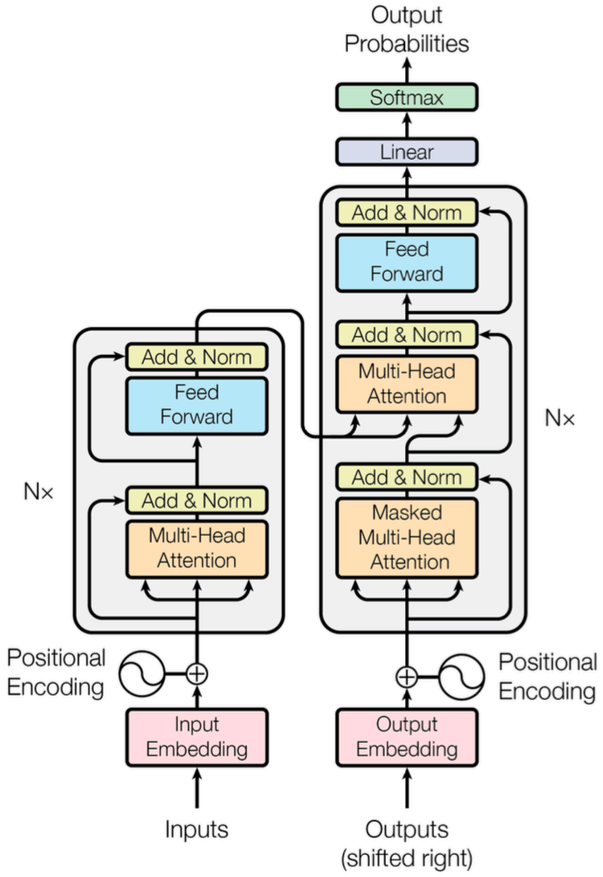
- **Backpropagation can also be modeled as a coalgebra X —> F(X)**
- **This alternative view gives us deeper insight into the convergence of backpropation**
- **It gives us more powerful tools to design new deep learning methods (see [Mahadevan, GAIA, Arxiv, 2024])**

Natural Transformations for Deep Learning





Category of Transformers



Objects: X is a n x d token sequence of matrices

Arrows: *permutation-equivariant* functions

[Yun et al., ICLR 2020]

f(XP) = f(X)P

$$\begin{array}{ccccc} X & \xrightarrow{f} & Y & \xrightarrow{g} & Z \\ \downarrow{\scriptstyle P} & & \downarrow{\scriptstyle P} & & \downarrow{\scriptstyle P} \\ XP & \xrightarrow[f]{} & YP & \xrightarrow[g]{} & ZP \end{array}$$

Commutative diagram

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

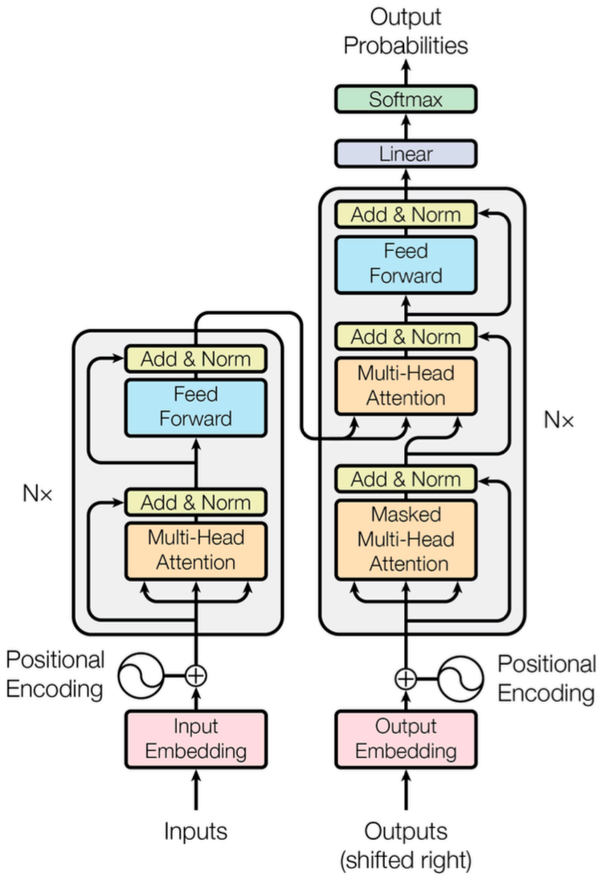
How do we define new Transformer architectures?

[Mahadevan, GAIA, Arxiv, 2024]

Enriched LLM Categories

- We can construct different types of categories for LLMs
  - Syntax category: L(x,y) = P(y | x) (where x =“I am flying”, y=“I am flying to Philadelphia”)
  - Semantics category: Yoneda Embedding: x —> L(x, -) (all possible completions of the phrase ``I am flying”)
  - Enriched category: the ``hom-object” L(x,y) is itself an object of another category (in this case, a symmetric monoidal preorder on [0,1]).

Intrinsic Limitations of Transformers



Theoretical Limitations of Self-Attention in Neural Sequence Models

Michael Hahn
Stanford University
mhahn2@stanford.edu

Abstract

Transformers are emerging as the new workhorse of NLP, showing great success across tasks. Unlike LSTMs, transformers process input sequences entirely through self-attention. Previous work has suggested that the computational capabilities of self-attention to process hierarchical structures are limited. In this work, we mathematically investigate the computational power of self-attention to model formal languages. Across both soft and hard attention, we show strong theoretical limitations of the computational abilities of self-attention, finding that it cannot model periodic finite-state languages, nor hierarchical structure, unless the number of layers or heads increases with input length. These limitations seem surprising given the practical success of self-attention and the prominent role assigned to hierarchical structure in linguistics, suggesting that natural language can be approximated well with models that are too weak for the formal languages typically assumed in theoretical linguistics.

1 Introduction

Transformers are emerging as the new workhorse of NLP, achieving the state-of-the-art in tasks such as language modeling, machine translation, and creating pretrained contextualized word embeddings. Eschewing recurrent computations, transformers are entirely based on self-attention, performing their computations largely in parallel

chical structure and recursion. Hierarchical structure is widely thought to be essential to modeling natural language, in particular its syntax (Everaert et al., 2015). Consequently, many researchers have studied the capability of recurrent neural network models to capture context-free languages (e.g., Kalinke and Lehmann (1998); Gers and Schmidhuber (2001); Grüning (2006); Weiss et al. (2018); Sennhauser and Berwick (2018); Korsky and Berwick (2019)) and linguistic phenomena involving hierarchical structure (e.g., Linzen et al. (2016); Gulordava et al. (2018)). Some experimental evidence suggests that transformers might not be as strong as LSTMs at modeling hierarchical structure (Tran et al., 2018), though analysis studies have shown that transformer-based models encode a good amount of syntactic knowledge (e.g., Clark et al. (2019); Lin et al. (2019); Tenney et al. (2019)).

In this work, we examine these questions from a theoretical perspective, asking whether models entirely based on self-attention are theoretically capable of modeling hierarchical structures involving unbounded recursion. Formally, we study their ability to perform two computations that are thought to be essential to hierarchical structure: First, their ability to correctly **close brackets**, a basic problem underlying all nonregular context-free languages and formalized by the DYCK language (Chomsky and Schützenberger, 1963). Second, their ability to **evaluate iterated negation**, a basic component of the task of evaluating logical

Published as a conference paper at ICLR 2023

NEURAL NETWORKS AND THE CHOMSKY HIERARCHY

Grégoire Delétang[*1] **Anian Ruoss**[*1] **Jordi Grau-Moya**[1] **Tim Genewein**[1] **Li Kevin Wenliang**[1]

Elliot Catt[1] **Chris Cundy**[†2] **Marcus Hutter**[1] **Shane Legg**[1] **Joel Veness**[1] **Pedro A. Ortega**[†]

ABSTRACT

Reliable generalization lies at the heart of safe ML and AI. However, understanding when and how neural networks generalize remains one of the most important unsolved problems in the field. In this work, we conduct an extensive empirical study (20 910 models, 15 tasks) to investigate whether insights from the theory of computation can predict the limits of neural network generalization in practice. We demonstrate that grouping tasks according to the Chomsky hierarchy allows us to forecast whether certain architectures will be able to generalize to out-of-distribution inputs. This includes negative results where even extensive amounts of data and training time never lead to any non-trivial generalization, despite models having sufficient capacity to fit the training data perfectly. Our results show that, for our subset of tasks, RNNs and Transformers fail to generalize on non-regular tasks, LSTMs can solve regular and counter-language tasks, and only networks augmented with structured memory (such as a stack or memory tape) can successfully generalize on context-free and context-sensitive tasks.

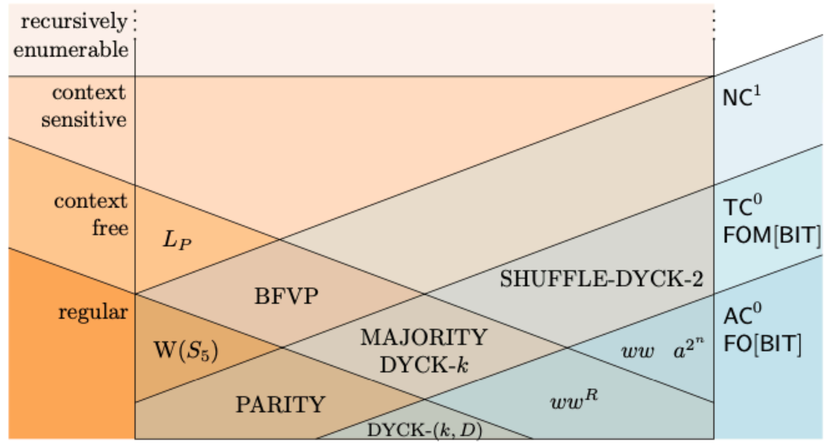
1 INTRODUCTION

Statistical learning theory is the most widely used theory of generalization in practical machine learning, justifying empirical risk minimization and estimating the generalization error via a test set (Vapnik, 1998). However, its central assumption that training and test data are independent and identically distributed (i.i.d.) is violated for many problems of interest (distribution shifts, continual learning, etc.). An example of such a non-i.i.d. setting is testing generalization on sequence prediction problems, where an agent is trained with sequences of length $\ell \leq N$ and tested with arbitrarily longer sequences $\ell \gg N$. This problem is of particular importance since it subsumes all computable problems (Dawid, 1984; Rich, 2007; Sipser, 1997; Solomonoff, 2009; 2010). Central to sequence prediction is *inductive inference*, which consists of deriving a general rule from a finite set of concrete instances and using this rule to make predictions. For example, in *program induction* (Goldberg, 1989; Gomez et al., 2008; Holland, 1992; Liang et al., 2013; Nordin, 1997; Solomonoff, 1964a;b; Wineberg & Oppacher, 1994), the goal is to obtain a model that correctly identifies the underlying data-generating process given examples of input-output sequences. Then, if the model is correct, it can produce results in accordance with the generative process for previously unseen input sequences.

"Hard-attention" Transformers cannot compute parity, recognize Dyck languages etc.

Computational Limitations of Transformers

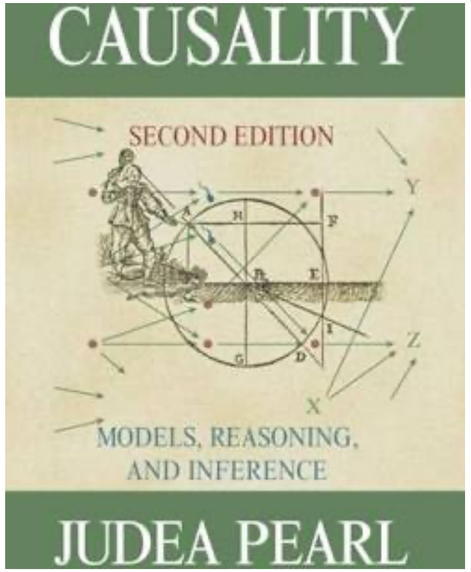
- Transformers can be modeled in terms of circuit complexity
  - AC^0, TC^0 : [Merrill et al., Saturated Transformers are Constant-Depth Threshold Circuits, 2022]
- Complexity Classes can be characterized as categories
  - Example: NP-complete problems define a category
  - Objects: NP-complete problem
  - Arrows: polynomial transformation from 3-SAT



[Strobl et al., What Formal Languages can Transformers Express, 2024]

Limitations of Deep Learning

- Deep learning models cannot reason about causality
  - A Transformer computes permutation-invariant functions
  - Attention scores are symmetric
  - Next-token distributions do not capture causality
- Generative models by themselves do not capture causality
  - “Curve-fitting” is not enough to do causal reasoning!



Universal Causality and Probability

The Atlantic

TECHNOLOGY

How a Pioneer of Machine Learning Became One of Its Sharpest Critics

Judea Pearl helped artificial intelligence gain a strong grasp on probability, but laments that it still can't compute cause and effect.

By Kevin Hartnett and Quanta Magazine

entropy

MDPI

Article

Universal Causality

Sridhar Mahadevan

Adobe Research, 345 Park Avenue, San Jose, CA 95110, USA; smahadev@adobe.com

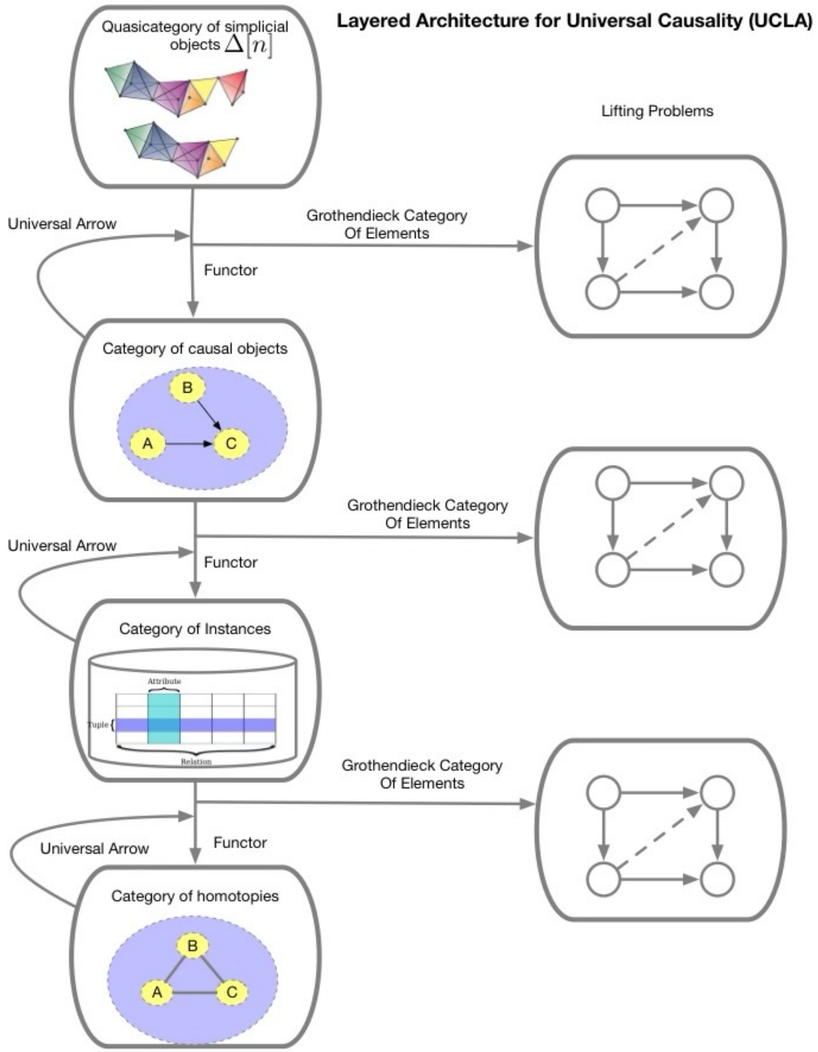
Abstract: Universal Causality is a mathematical framework based on higher-order category theory, which generalizes previous approaches based on directed graphs and regular categories. We present a hierarchical framework called UCLA (Universal Causality Layered Architecture), where at the top-most level, causal interventions are modeled as a higher-order category over simplicial sets and objects. Simplicial sets are contravariant functors from the category of ordinal numbers Δ into sets, and whose morphisms are order-preserving injections and surjections over finite ordered sets. Non-random interventions on causal structures are modeled as face operators that map n-simplices into lower-level simplices. At the second layer, causal models are defined as a category, for example defining the schema of a relational causal model or a symmetric monoidal category representation of DAG models. The third layer corresponds to the data layer in causal inference, where each causal object is mapped functorially into a set of instances using the category of sets and functions between sets. The fourth homotopy layer defines ways of abstractly characterizing causal models in terms of homotopy colimits, defined in terms of the nerve of a category, a functor that converts a causal (category) model into a simplicial object. Each functor between layers is characterized by a universal arrow, which define universal elements and representations through the Yoneda Lemma, and induces a Grothendieck category of elements that enables combining formal causal models with data instances, and is related to the notion of *ground graphs* in relational causal models. Causal inference between layers is defined as a lifting problem, a commutative diagram whose objects are categories, and whose morphisms are functors that are characterized as different types of fibrations. We illustrate UCLA using a variety of representations, including causal relational models, symmetric monoidal categorical variants of DAG models, and non-graphical representations, such as integer-valued multisets and separoids, and measure-theoretic and topological models.

Keywords: artificial intelligence; higher-order category theory; causality; machine learning; statistics

Citation: Mahadevan. S. Universal

Table 2. Each layer of UCLA represents a categorical abstraction of causal inference.

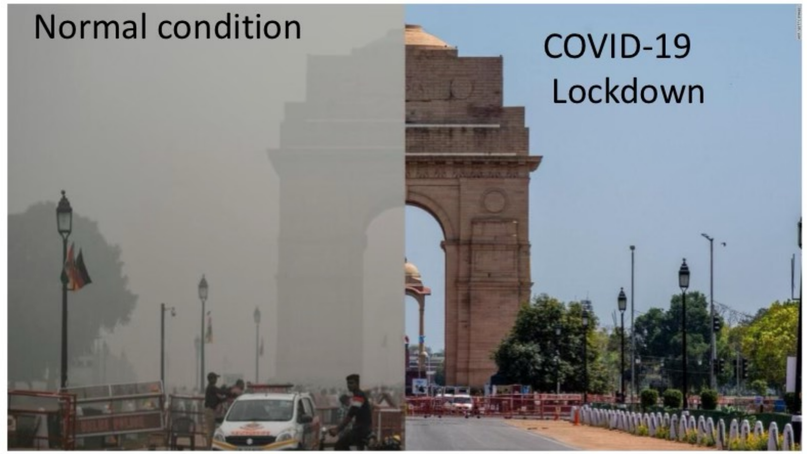
Layer	Objects	Morphisms	Description
Simplicial	$[n] = \{0, 1, \ldots, n\}$	$f = [m] \to [n]$	Category of interventions
Relational	Vertices V, Edges E	$s, t : E \to V$	Causal Model Category
Tabular	Sets	Functions on sets $f : S \to T$	Category of instances
Homotopy	Topological Spaces	Causal equivalence	Causal homotopy

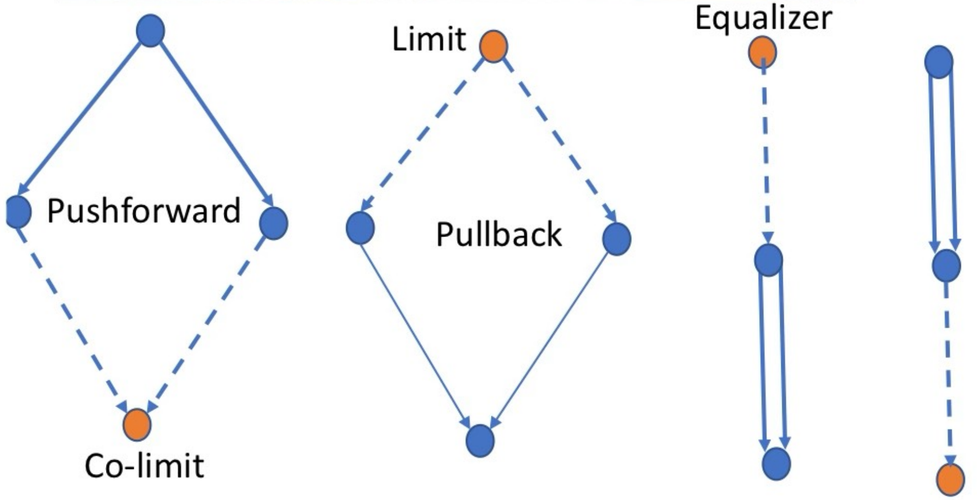


Universal Causality

[Mahadevan, Entropy, 2023]

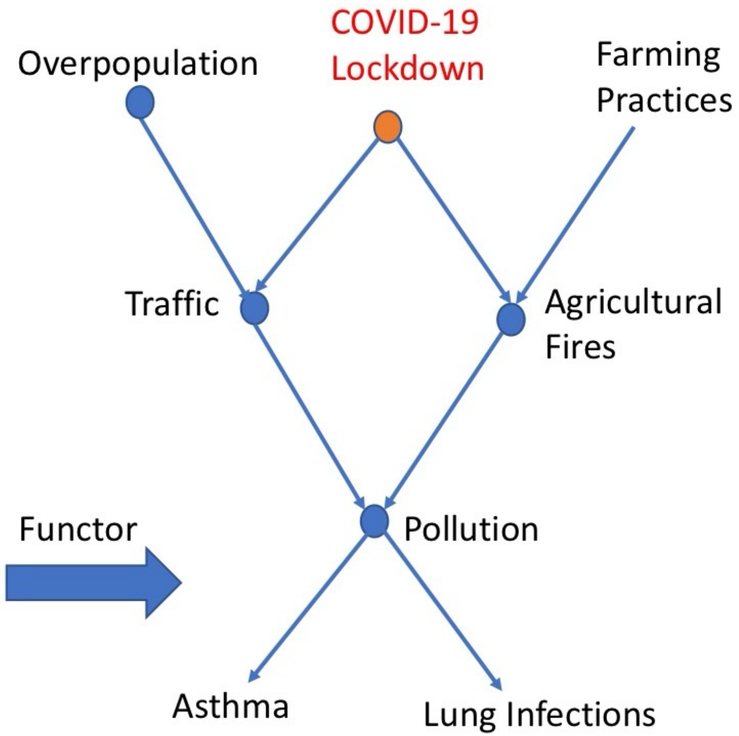
Pollution in New Delhi, India





Indexing Category of Abstract Diagrams

Functor



Actual Causal Model

What is categorical probability?

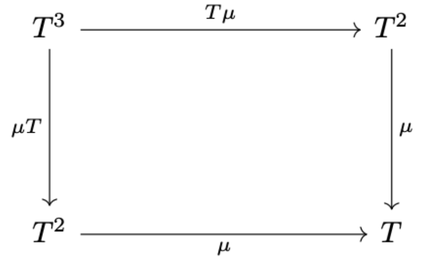
- 18th century: Bernoulli, Laplace: coin-tossing, card games
- 20th century: Kolmogorov: measure-theoretic foundations
- 21st century: Probabilities are monads!
- String diagrams are sufficient for statistical reasoning!

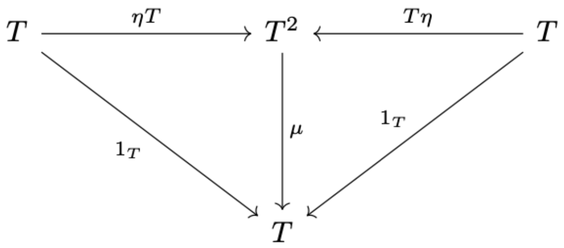
Category of Structural Causal Models

- A structural causal model (SCM) is defined as M = (U, V, F)
  - U is a set of exogenous causal variables
  - V is a set of endogenous variables defined over U and V
  - F is a collection of ``local autonomous functions" f_i: U x V/V_i —> V_i
- Objects: SCM models M, M', …
- Arrows: A function M —> M' that represents an intervention etc.
- How to ensure local functions define a unique global function F?

Monads

T: C -> C (endofunctor)





Example: T = 2^X (powerset)

Giri monad: probability

$$\Pi : \mathrm{Meas} \to \mathrm{Meas}$$

$$\mathrm{ev}_A : \Pi(.) \mapsto [0,1]$$
$$\mathbb{P} \mapsto \mathbb{P}(A)$$

$$A \mapsto \int \mathrm{ev}_A d\mathbf{P}$$

This functor maps a measurable space X to all

distributions on X (which is also measurable!)

A synthetic approach to Markov kernels, conditional independence and theorems on sufficient statistics

Tobias Fritz

Abstract. We develop *Markov categories* as a framework for synthetic probability and statistics, following work of Golubtsov as well as Cho and Jacobs. This means that we treat the following concepts in purely abstract categorical terms: conditioning and disintegration; various versions of conditional independence and its standard properties; conditional products; almost surely; sufficient statistics; versions of theorems on sufficient statistics due to Fisher–Neyman, Basu, and Bahadur.

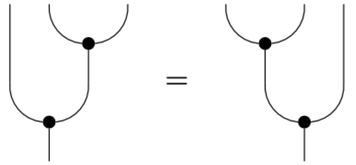
Besides the conceptual clarity offered by our categorical setup, its main advantage is that it provides a uniform treatment of various types of probability theory, including discrete probability theory, measure-theoretic probability with general measurable spaces, Gaussian probability, stochastic processes of either of these kinds, and many others.

Contents

2.1. Definition. *A* Markov category C *is a symmetric monoidal category in which every object* $X \in \mathsf{C}$ *is equipped with a commutative comonoid structure given by a comultiplication* $\mathsf{copy}_X : X \to X \otimes X$ *and a counit* $\mathsf{del}_X : X \to I$*, depicted in string diagrams as*

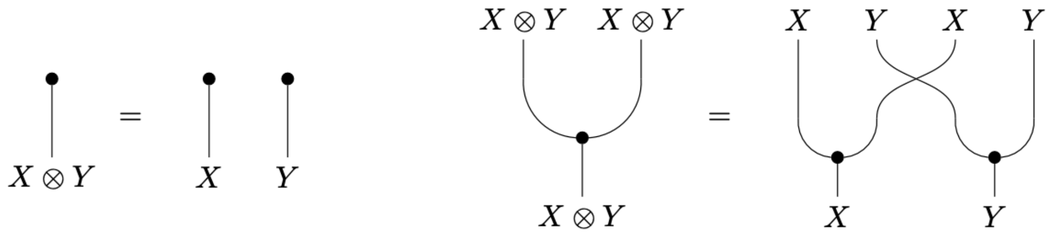
copy_X =  del_X = (2.1)

and satisfying the commutative comonoid equations,

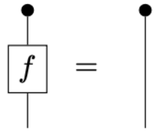
 (2.2)

 (2.3)

as well as compatibility with the monoidal structure,

 (2.4)

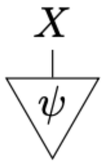
and naturality of del*, which means that*

 (2.5)

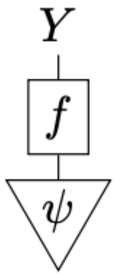
for every morphism f*.*

Probabilistic Reasoning with Diagrams

$\psi : I \to X$

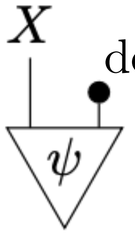


Distribution



$f : X \to Y$

Random Variable



$\mathrm{del}_Y : Y \to I$

$\psi : I \to X \otimes Y$

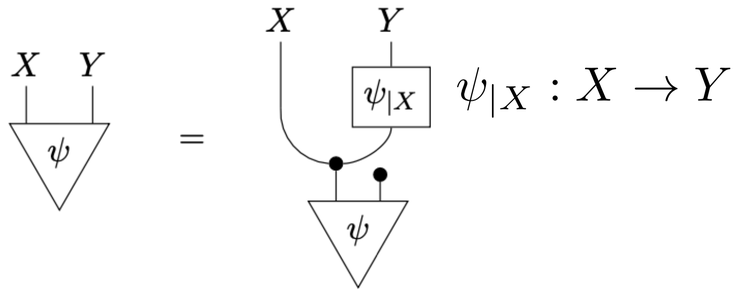
Marginalization

Replaces the conventional ``measure-theoretic" foundation of probability

[Fritz, Adv. in Math, 2020]

String Diagrams: Markov Categories

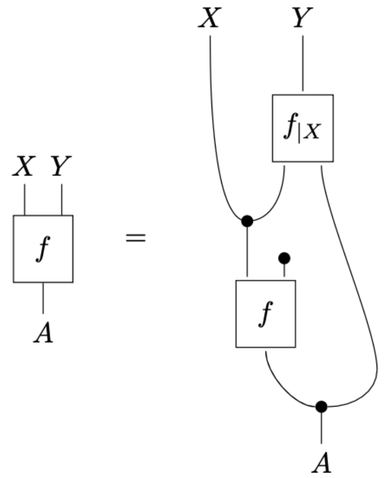
$\psi : I \to X \otimes Y$



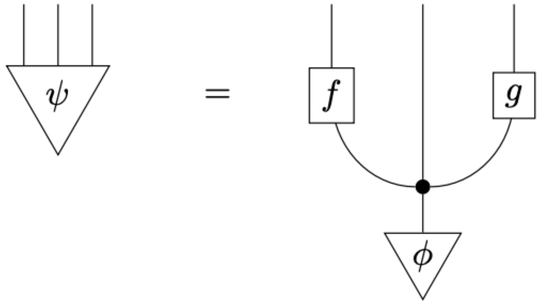
$\psi_{|X} : X \to Y$

Conditional Distribution

Generalizes P(X,Y) = P(Y|X) P(X)



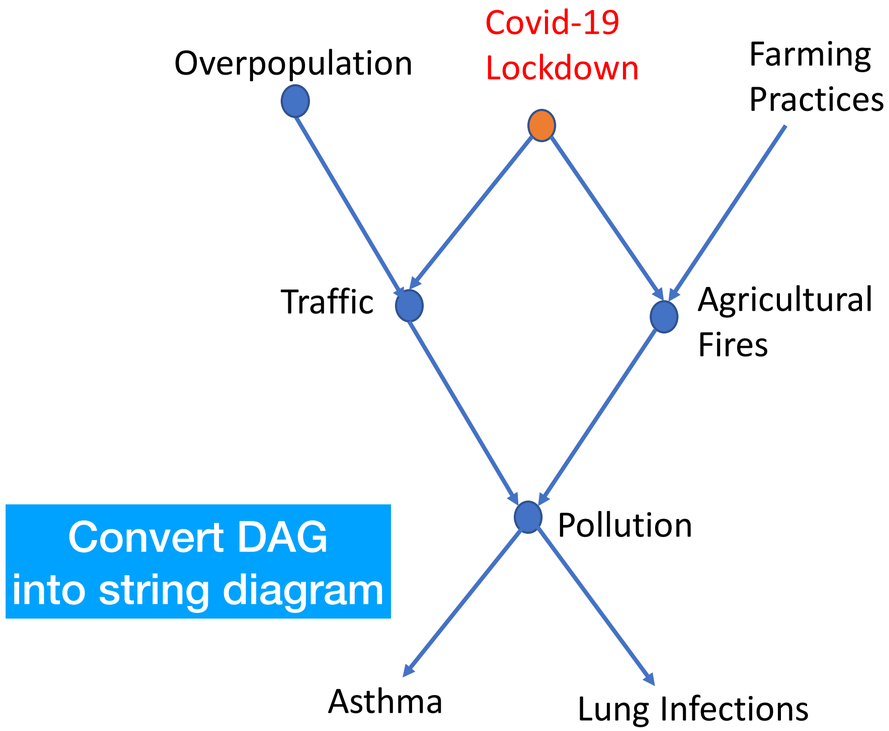
Disintegration:Bayes Rule



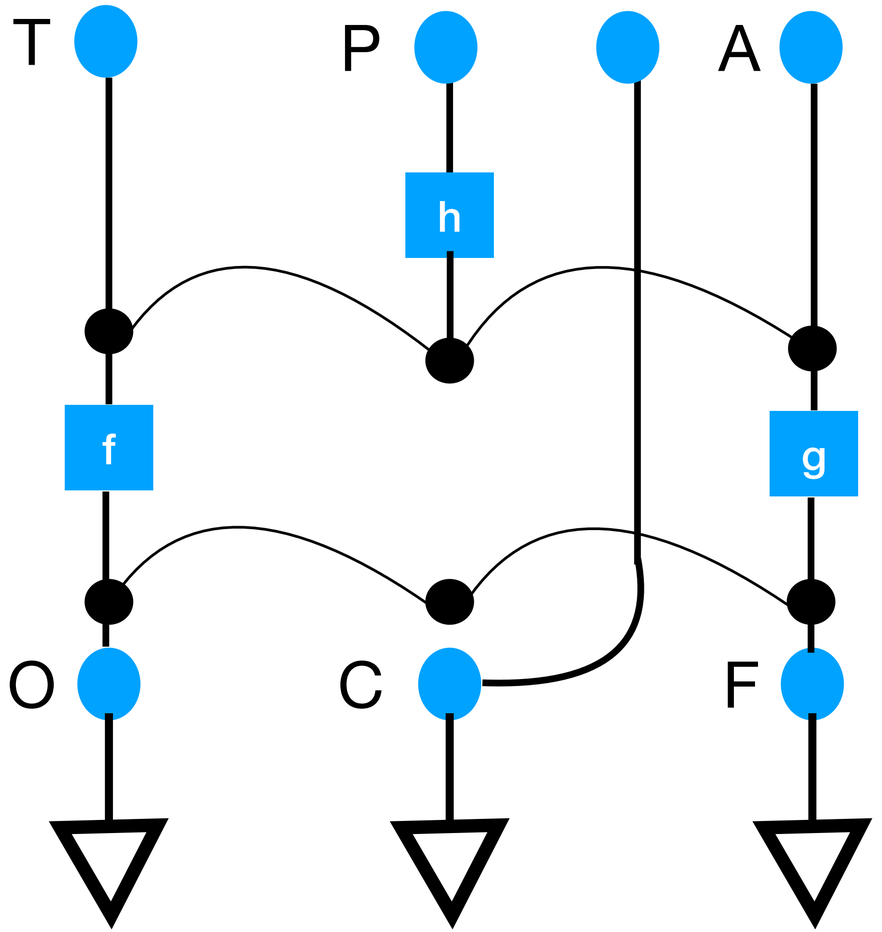
Conditional independence

P(X, W, Y) = P(X | W) P(Y | W)

Causal Models as String Diagrams

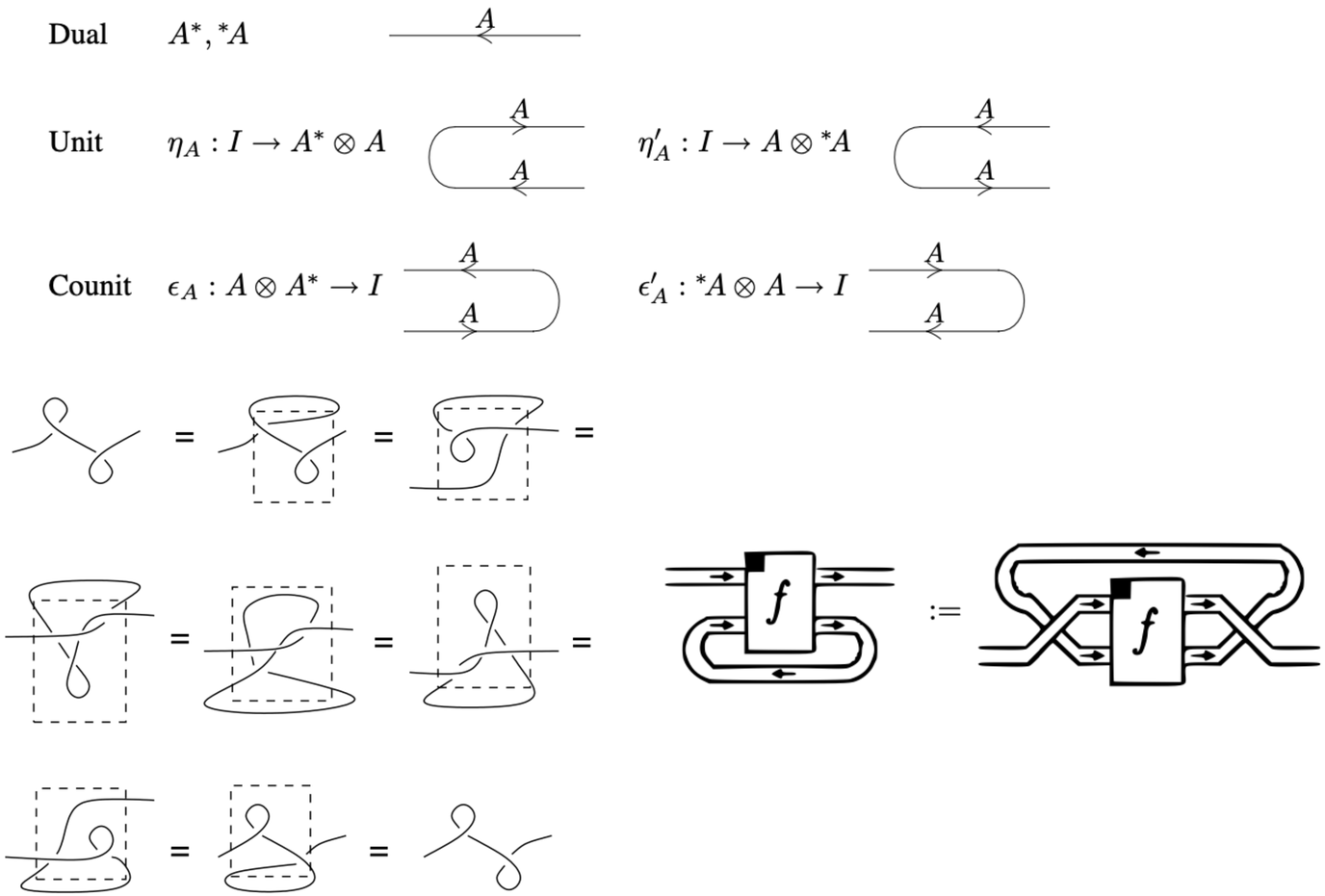


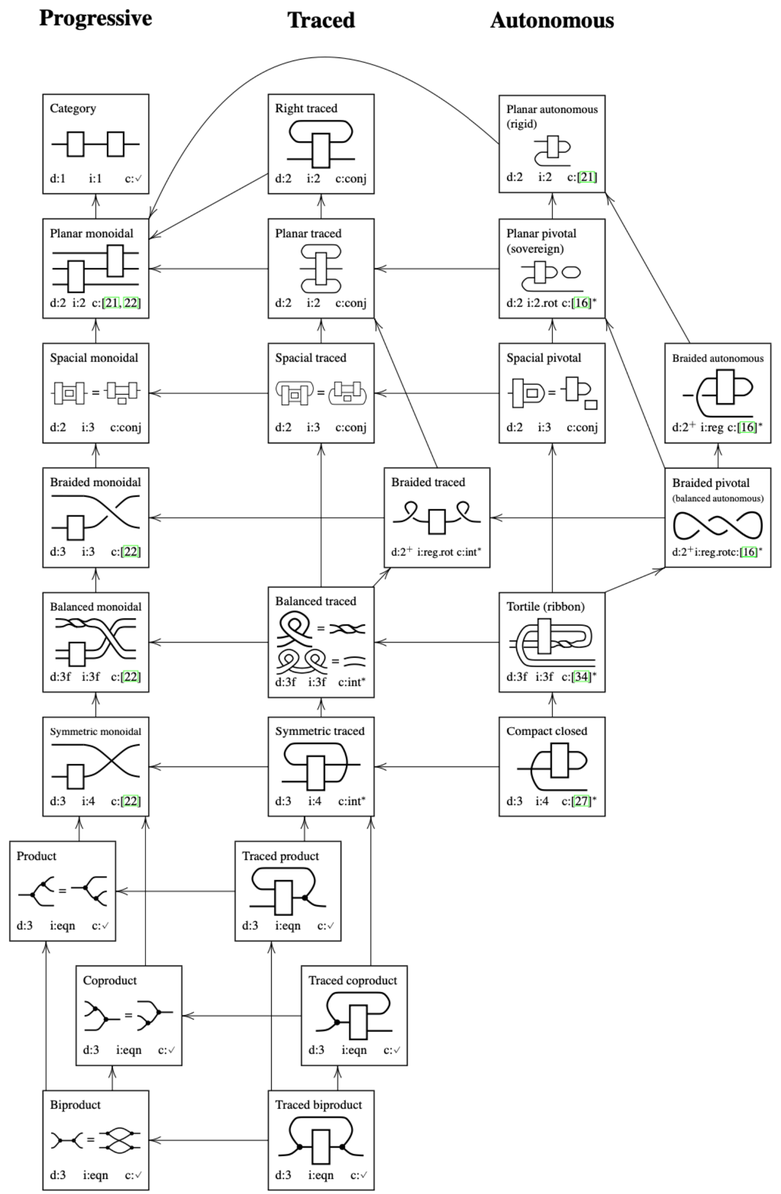
Convert DAG into string diagram



Exogenous variables

Rich Family of String Diagrams





[Selinger, A survey of graphical languages for monoidal categories, Arxiv, 2020]

Categorical Knowledge Representation

How do we represent knowledge?

- A popular formalism for representing knowledge is through relations
- Relational databases is the most used format in industry to store information
- A relational schema defines a category
- SQL query languages can be formalized as lifting diagrams in categories

Relational Database

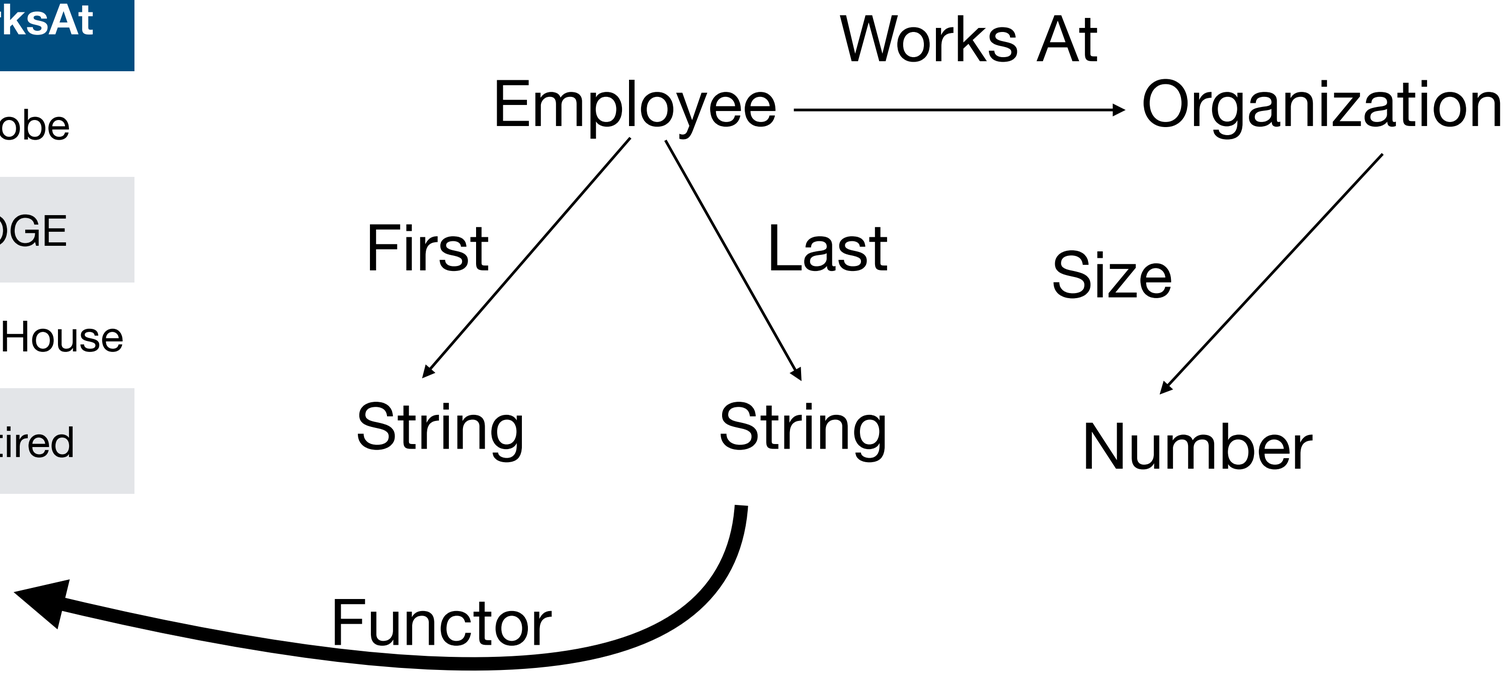
Person	First	Last	Resident
ID1	Sridhar	Mahadevan	California
ID2	Elon	Musk	Texas
ID3	Donald	Trump	DC
ID4	Joe	Biden	Delaware

Person	Education	Martial Status	Children
ID1	PhD	Married	None
ID2	BS	Married	Yes
ID3	BS	Married	Yes
ID4	BA	Married	Yes

Person	Nationality	Born In	Primary Home
ID1	USA	India	CA
ID2	USA	South Africa	Texas
ID3	USA	USA	Florida
ID4	USA	USA	Delaware

Databases are Categories

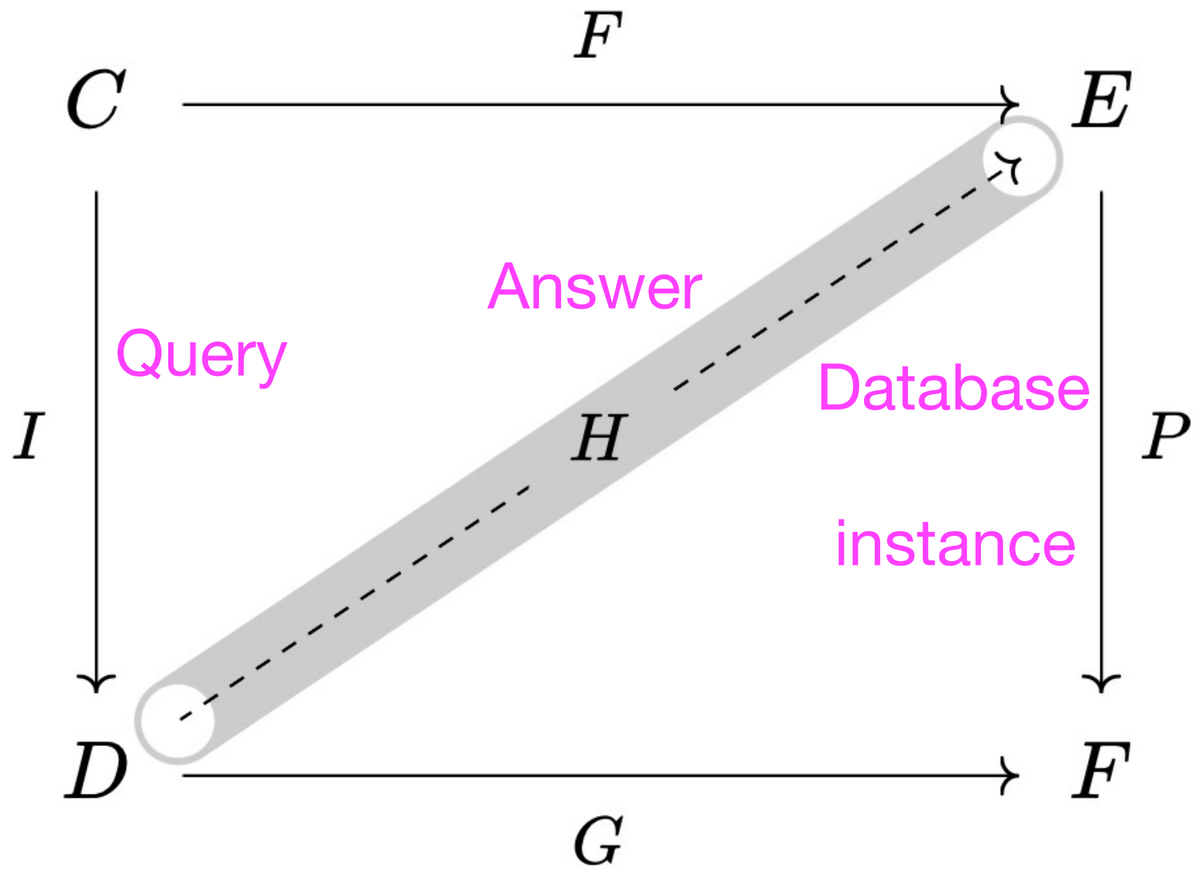
ID	First	Last	WorksAt
ID1	Sridhar	Mahadevan	Adobe
ID2	Elon	Musk	DOGE
ID3	Donald	Trump	WhiteHouse
ID4	Joe	Biden	Retired



Find a White House employee who was born in South Africa

Find an Adobe employee born in India

Database queries are Lifting Diagrams



$$F = H \circ I$$

$$G = H \circ P$$

$$G \circ I = P \circ F$$

[Spivak, Database Queries and Constraints via Lifting Problems, 2013]

Open Games

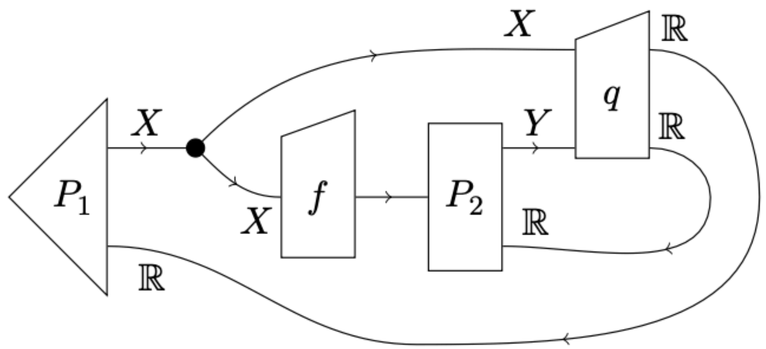
Game Theory

- Developed by von Neumann and Nash as a model of economics
- Widely applied in auctions of multi-billion dollar commodities (spectrum)
- Basis behind a lot of e-commerce
- Several Nobel prizes have been awarded in game theory
- But: game theory is not compositional

Prisoner’s Dilemma

Players	Confess	Lie
Confess	(-10, -10)	(0, -10)
Lie	(-10,0)	(-1, -1)

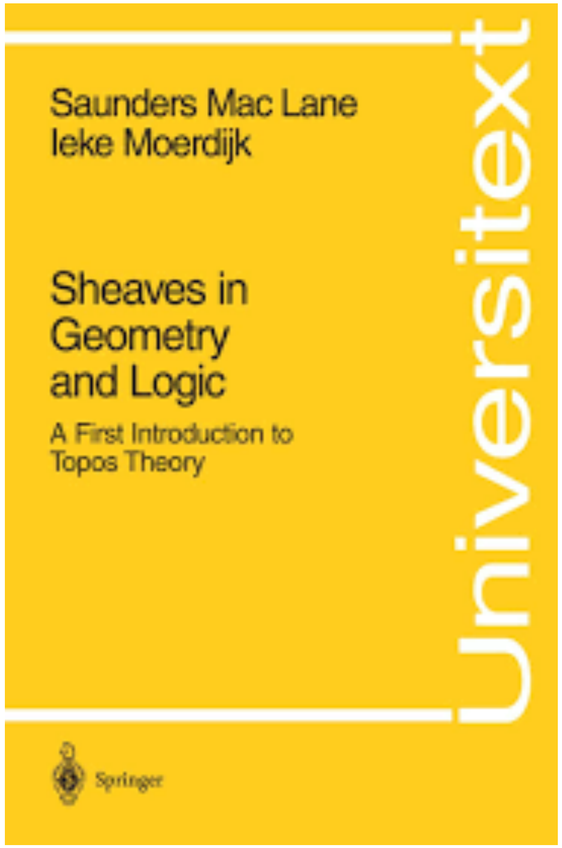
Compositional game theory



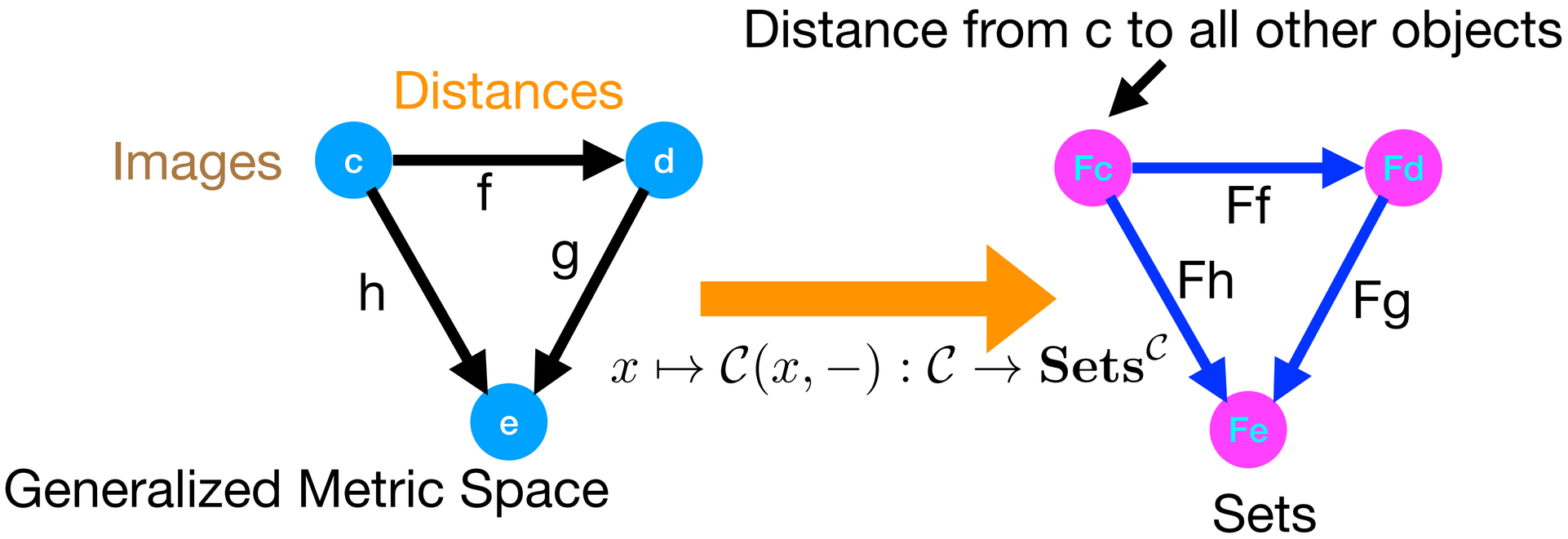
- The category of open games is defined as a symmetric monoidal category
- It uses the concept of lenses
- Just like the abstract deep learning model, we need a ``request" function
- In open games, it is called coplay

[Hedges, "Morphisms of Open Games", Electr. Notes in TCS, 2018]

Functor Categories



Set-Valued Functors



Yoneda embeddings show how to construct generalized representers

Functor Categories

- The most interesting categories are those where every object is a functor!
  - Presheaves are constructed through the Yoneda embedding
  - Simply map an object x —> C(-, x)
- The most beautiful embedding in all of mathematics!
  - Yoneda embeddings create a nicer ``copy” of a category
  - The category of presheaves is a topos

Contravariant and Covariant Functor Categories

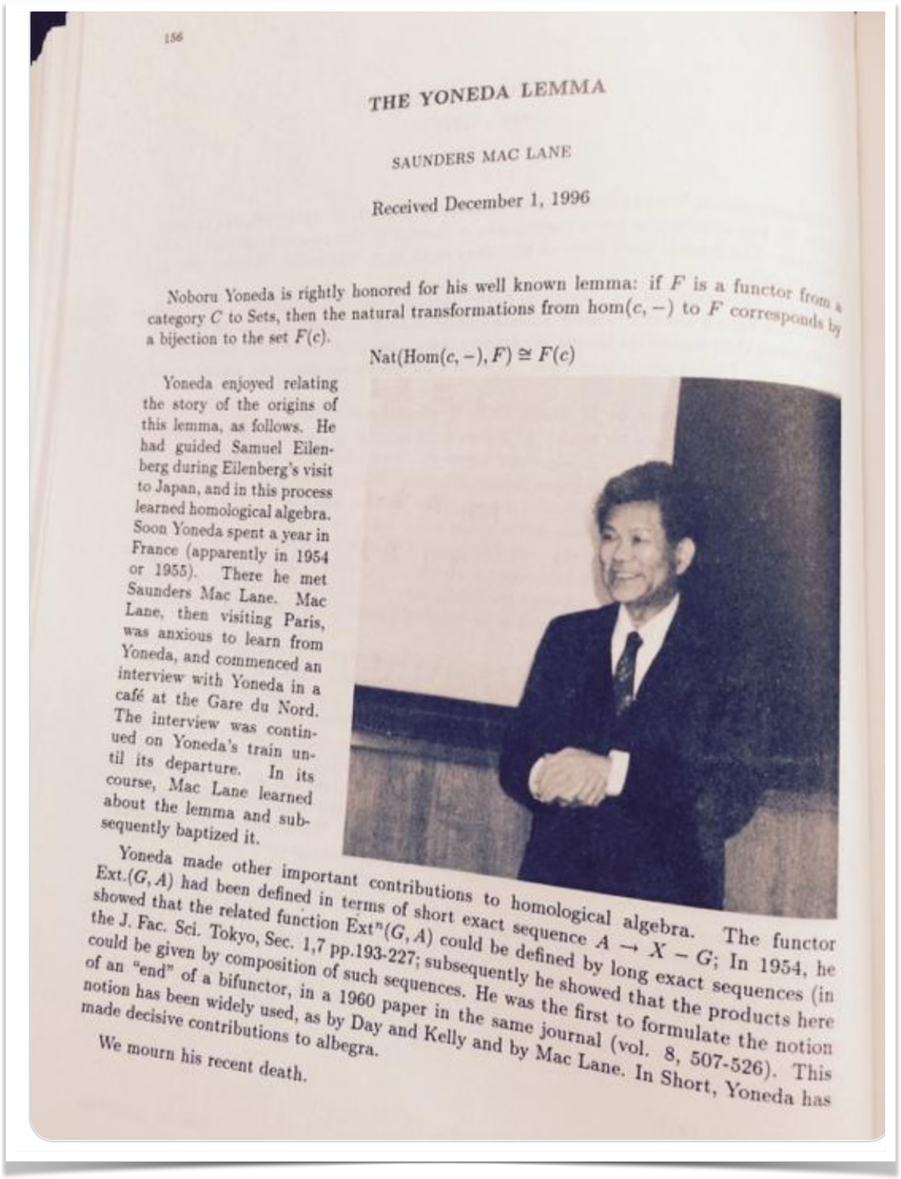
$$x \mapsto \mathcal{C}(-, x) : \mathcal{C} \to \mathbf{Sets}^{\mathcal{C}^{op}}$$

Presheaves: Contravariant

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

Copresheaves: Covariant

Yoneda Lemma



156

THE YONEDA LEMMA

SAUNDERS MAC LANE

Received December 1, 1996

Noboru Yoneda is rightly honored for his well known lemma: if F is a functor from a category C to Sets, then the natural transformations from $\hom(c, -)$ to F corresponds by a bijection to the set $F(c)$.

$$\mathrm{Nat}(\mathrm{Hom}(c, -), F) \cong F(c)$$

Yoneda enjoyed relating the story of the origins of this lemma, as follows. He had guided Samuel Eilenberg during Eilenberg's visit to Japan, and in this process learned homological algebra. Soon Yoneda spent a year in France (apparently in 1954 or 1955). There he met Saunders Mac Lane. Mac Lane, then visiting Paris, was anxious to learn from Yoneda, and commenced an interview with Yoneda in a café at the Gare du Nord. The interview was continued on Yoneda's train until its departure. In its course, Mac Lane learned about the lemma and subsequently baptized it.

Yoneda made other important contributions to homological algebra. The functor $\mathrm{Ext}_*(G, A)$ had been defined in terms of short exact sequence $A \to X - G$; In 1954, he showed that the related function $\mathrm{Ext}^n(G, A)$ could be defined by long exact sequences (in the J. Fac. Sci. Tokyo, Sec. 1,7 pp.193-227; subsequently he showed that the products here could be given by composition of such sequences. He was the first to formulate the notion of an "end" of a bifunctor, in a 1960 paper in the same journal (vol. 8, 507-526). This notion has been widely used, as by Day and Kelly and by Mac Lane. In Short, Yoneda has made decisive contributions to albegra.

We mourn his recent death.

$$\mathrm{Nat}(C(-, x), F) \simeq Fx$$

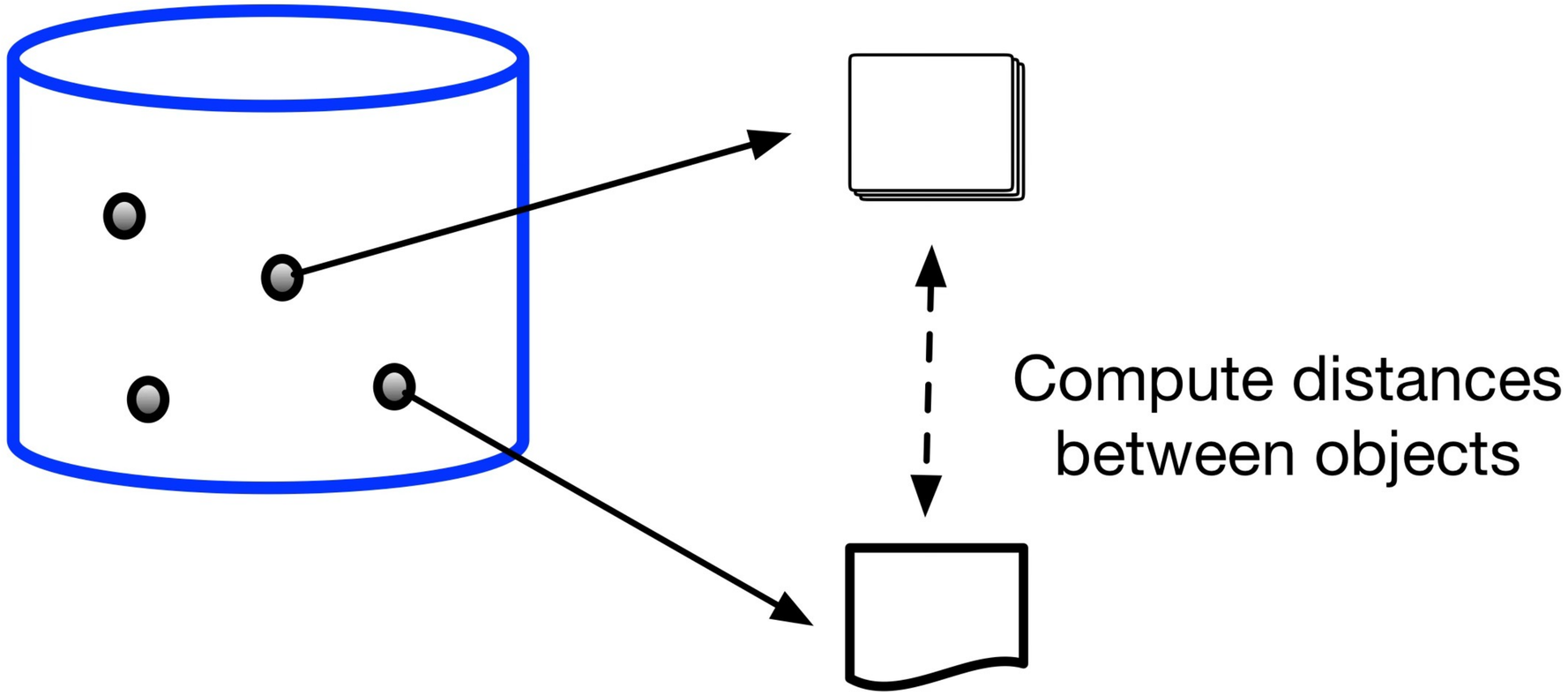
"Objects are defined by their interactions"

The Yoneda Lemma came to "life" in 1954

Coincidentally, Turing died in 1954

Metric Yoneda Lemma in Generalized Metric Spaces

Images, Text documents,
Probability Distributions…



ELSEVIER

Theoretical Computer Science 193 (1998) 1–51

Theoretical Computer Science

Fundamental Study

Generalized metric spaces: Completion, topology, and powerdomains via the Yoneda embedding

M.M. Bonsangue[a,*], F. van Breugel[b], J.J.M.M. Rutten[c]

[a] *Rijks Universiteit Leiden, Department of Computer Science, P.O. Box 9512, 2300 RA Leiden, The Netherlands*

[b] *Università di Pisa, Dipartimento di Informatica, Corso Italia 40, 56125 Pisa, Italy*

[c] *CWI, P.O. Box 94079, 1090 GB Amsterdam, The Netherlands*

Received June 1996; revised January 1997

Communicated by M. Nivat

Abstract

Generalized metric spaces are a common generalization of preorders and ordinary metric spaces (Lawvere, 1973). Combining Lawvere's (1973) enriched-categorical and Smyth's (1988, 1991) topological view on generalized metric spaces, it is shown how to construct (1) completion, (2) two topologies, and (3) powerdomains for generalized metric spaces. Restricted to the special cases of preorders and ordinary metric spaces, these constructions yield, respectively: (1) chain completion and Cauchy completion; (2) the Alexandroff and the Scott topology, and the ε-ball topology; (3) lower, upper, and convex powerdomains, and the hyperspace of compact subsets. All constructions are formulated in terms of (a metric version of) the Yoneda (1954) embedding.

Generalized Metric Spaces

- A generalized metric space (gms) is defined as a space X, where
  - $X(x, y) : X \times X \rightarrow [0,\infty]$
  - $X(x, x) = 0$
  - Triangle inequality: $X(x, z) \leq X(x, y) + X(y, z)$
  - Note: in a gms, symmetry does not hold, and two objects that are at distance 0 need not be identical

Examples: gms over Preorders

- Let us define a gms over a preordered set $(P, \leq)$
  - Reflexivity: $x \leq x$
  - Transitivity: $x \leq y, y \leq z \Rightarrow x \leq z$
  - The gms is defined as
    - If $x \leq y$, then $P(x, y) = 0$
    - If $x \not\leq y$, then $P(x, y) = \infty$

Example: gms over strings

- Consider the set of strings Σ^* over some alphabet Σ
- We can define a gms over the strings Σ^* as follows:
  - $\Sigma^*(x, y) = 0$ if x is a prefix of y
  - $\Sigma^*(x, y) = 2^{-n}$ otherwise where n is the longest common prefix

Example: gms over topological spaces

- We can define a gms over the power set $\mathscr{P}(X)$ of all subsets over a metric space as:
  - $\mathscr{P}(X)(V, W) = \inf\left(\epsilon > 0 \,|\, \forall v \in V, \exists w \in W \text{ s.t. } X(v, w) \leq \epsilon\right)$
- This distance is referred to as the non-symmetric Hausdorff distance

Example: gms over distances

- Let us define a gms over the category $[0,\infty]$ of non-negative distances:
  - $[0,\infty](x, y) = 0$ if $x \geq y$
  - $[0,\infty](x, y) = y - x$ if $x < y$
- This category is complete and co-complete, symmetric monoidal, as well as compact and closed
  - Product of two elements is their max (or supremum)
  - Coproducts of two elements is their minimum (or infimum)
  - Monoidal product is defined as addition +

Compact Closed Categories

- Let us define an “internal” Hom functor $[0,\infty](-,-)$ as simply the distance in $[0,\infty]$ as given previously
- The Yoneda embedding $[0,\infty](t,-)$ is ***right adjoint*** to $t+-$ for any $t \in [0,\infty]$
- **Theorem:** For all $r, s, t \in [0,\infty]$,
  - $t+s \geq r$ if and only if $s \geq [0,\infty](t,r)$

Metric Yoneda Lemma for gms

- We can construct “universal representers” in any gms by applying the Yoneda Lemma
- Let X be any gms. For any element $x \in X$
  - $X(-, x) : X^{op} \to [0,\infty] : y \mapsto X(y, x)$
- Let us define a category over gms by using as arrows all non-expansive functions f
  - $Y(f(x), f(y)) \leq c \cdot X(x, y)$
  - Where $c \in (0,1)$

Presheaves in a gms

- For any category C, define its presheaf $\hat{C} = \mathrm{Set}^{C^{op}}$
- In particular, the presheaf for the category of gms is given as
  - $\hat{X} = [0,\infty]^{X^{op}}$
  - Which defines the set of all non-expansive functions from X^{op} to $[0,\infty]$
  - Remarkably, the Yoneda embedding $y \mapsto [0,\infty](y,x)$ is itself a non-expansive mapping, and therefore an element of $\hat{X}$

Metric Yoneda Lemma

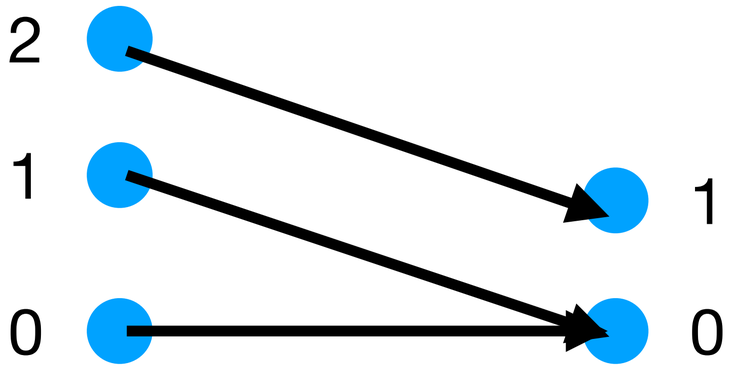
- For any non-expansive function $\phi \in \hat{X}$
  - $\hat{X}(X(-, x), \phi) = \phi(x)$
- The Yoneda embedding is an *isometry!*
  - $\mathbf{y}(x) = X(-, x)$
  - $X(x, y) = \hat{X}(\mathbf{y}(x), \mathbf{y}(y)) = \hat{X}(X(-, x), X(-, y))$
- Recall we have made no assumptions about symmetry!

Non-symmetric Attention in LLMs

- Recall that Tranformer modules compute permutation-equivariant maps because attention matrices are symmetric!
- To fix that problem, a Transformer uses Absolute Positional Encoding
- But, that “fix” causes problems of generalization in long sequences
- Conjecture: Yoneda embeddings in a gms may lead to new insights into attention in LLMs

Simplicial Category Δ

- **Objects**: ordinal numbers
  - $[n] = \{0,1,\ldots,n-1\}$
- **Arrows**:
  - $f : [m] \to [n]$
  - If $i \leq j$, then $f(i) \leq f(j)$

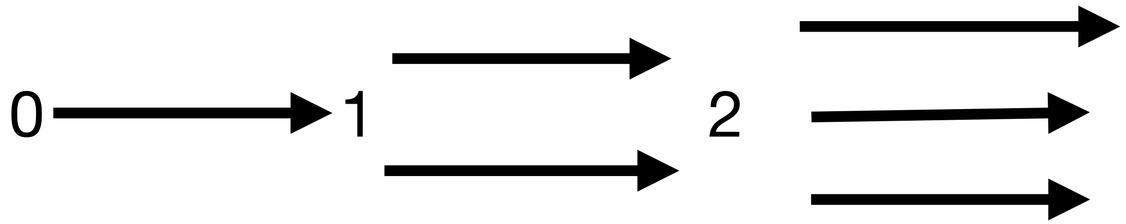


  - All morphisms can be built out of primitive injections/surjections
    - $\delta_i : [n] \to [n+1]$: injection skipping i
    - $\sigma_i : [n] \to [n-1]$, surjection repeating i

Nerve of a Category

- Recall a category is defined as a collection of objects, and a collection of arrows between any pair of objects
- A simplicial set is a contravariant functor mapping the simplicial category to the category of sets
- Any category can be mapped onto a simplicial set by constructing its nerve
- Intuitively, consider all sequences of composable morphisms of length n!

Simplicial Sets: Contravariant Functors



$$\delta_i^n : [n] \to [n+1]$$

$$\sigma_i^n : [n+1] \to [n]$$

$$X_n : [n] \to X : \Delta^{op} \to X$$

Simplicial Sets vs. Categories

- Any category can be embedded faithfully into a simplicial set using its nerve
- The embedding is full and faithful (perfect reconstruction)
- Unfortunately, the converse is not possible
- Given a simplicial set, the left adjoint functor that maps it into a category is lossy!
- GAIA (in theory!) is more powerful than existing generative AI formalisms

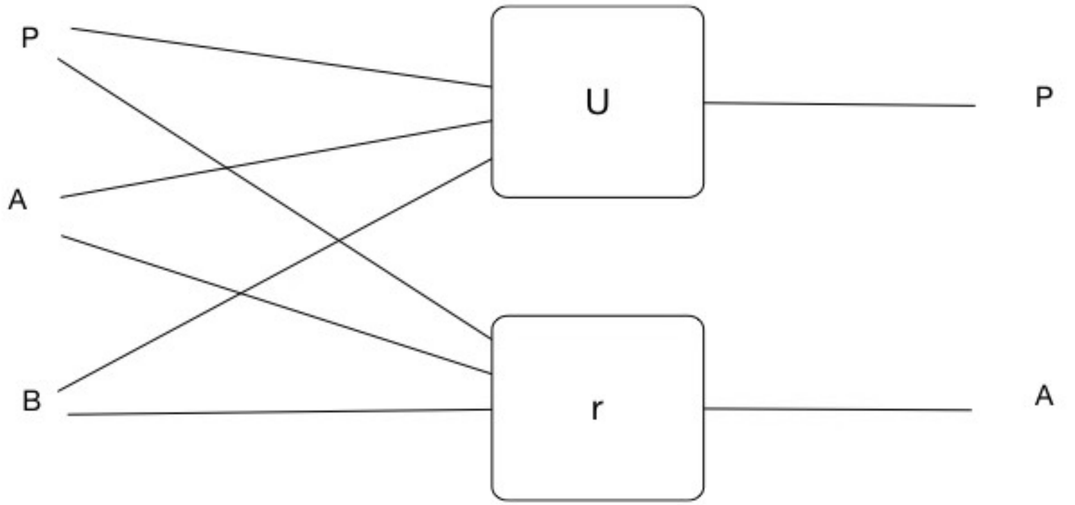
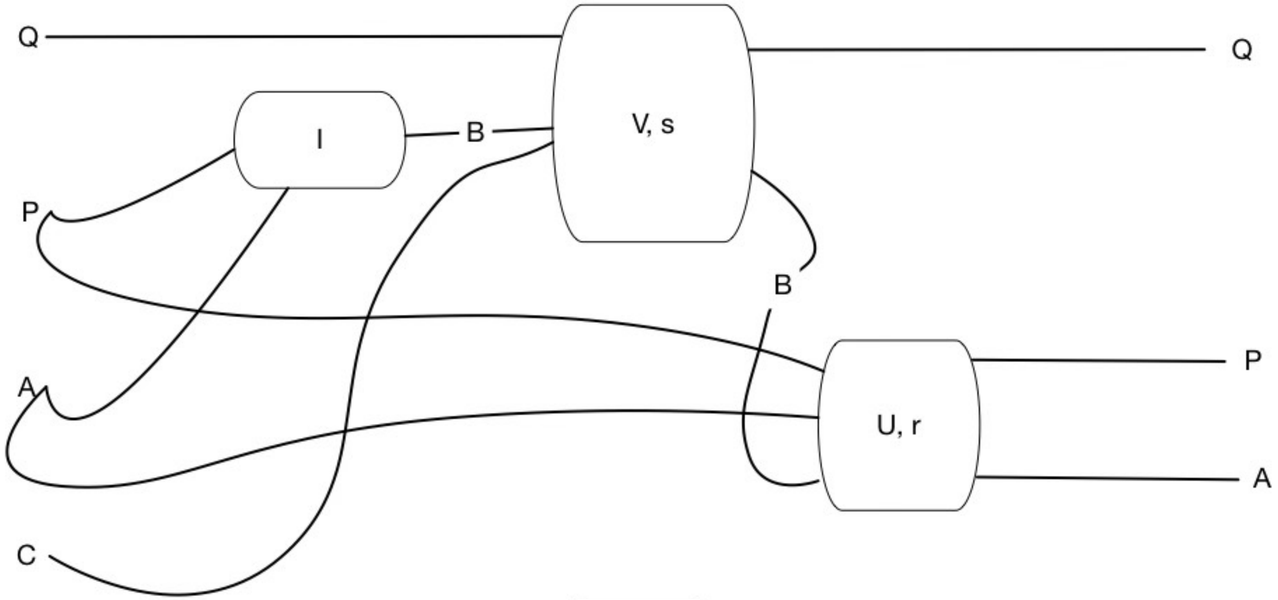


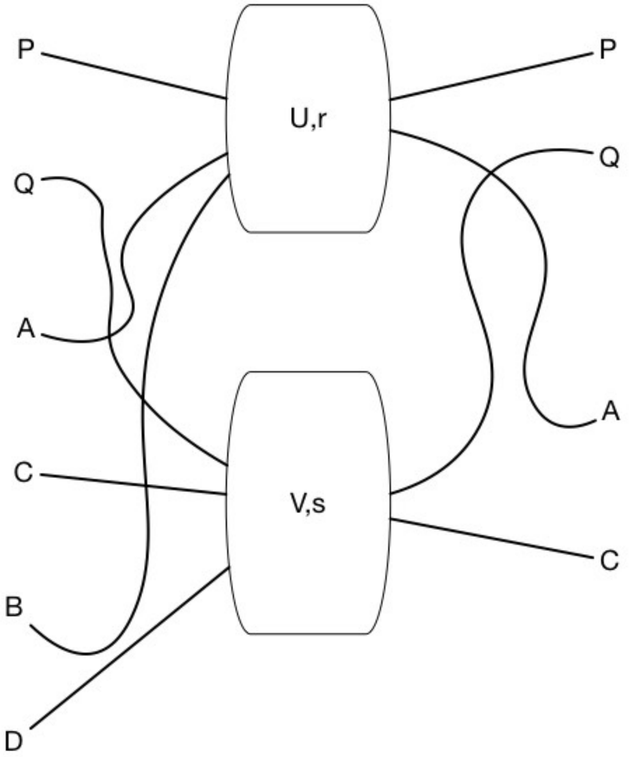
Figure 10: A learner in the symmetric monoidal category `Learn` is defined as a morphism. Later in Section 3, we will see how to define learners as coalgebras instead.

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

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



Sequential Composition



Parallel Composition

How to compose a sequence of two Learner objects?

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

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

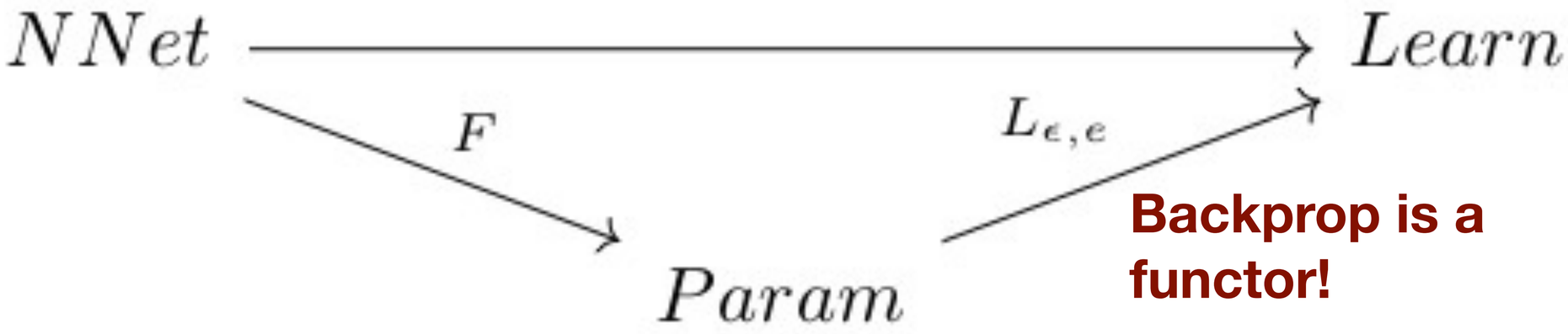
$$(I \cdot J)(p, q, a) \coloneqq J(q, I(p, a))$$

and the composite update function is

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

and the composite request function is

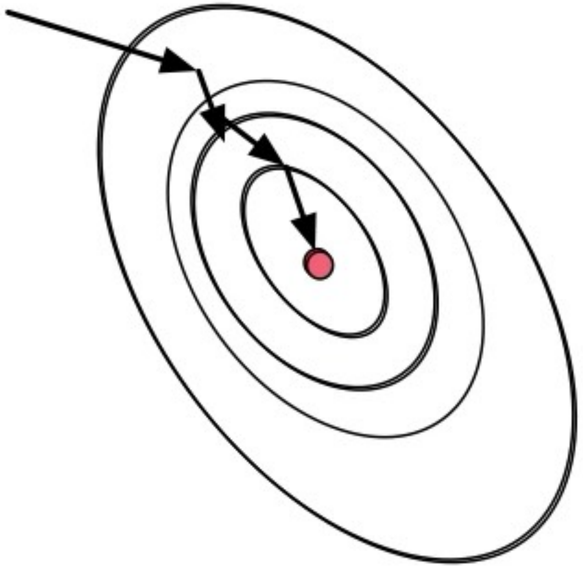
$$(r \cdot s)(p, q, a, c) \coloneqq r(p, a, s(q, I(p, a), c)).$$

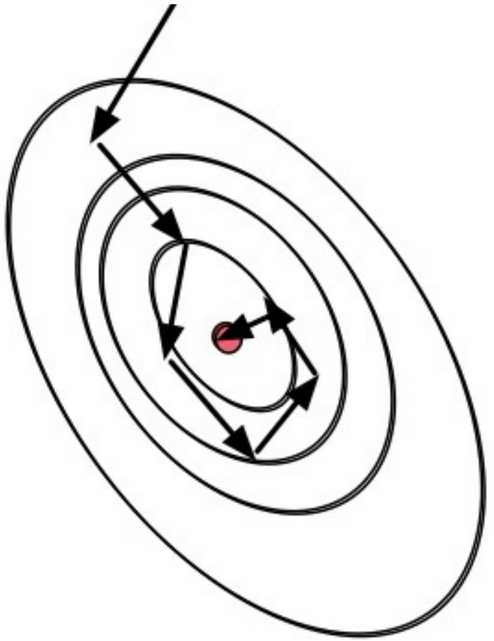


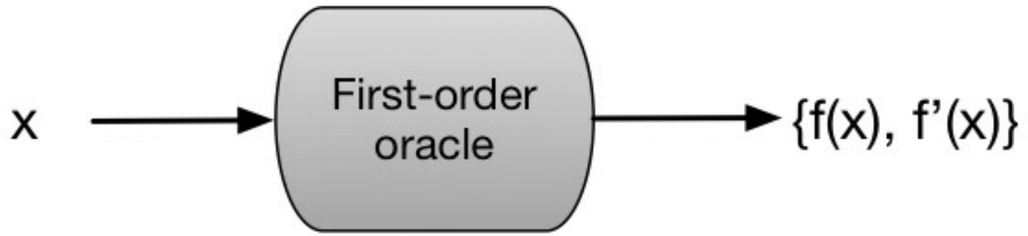
Backprop is a functor!

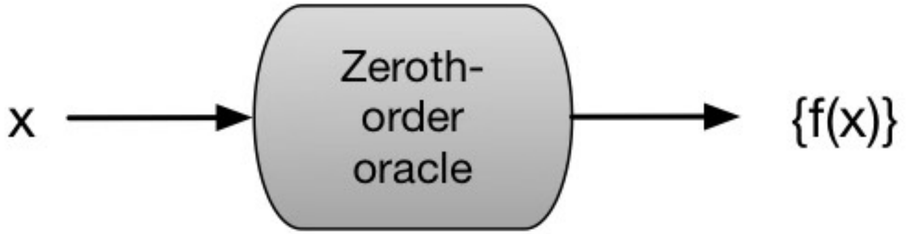
$$U_I(p, a, b) \coloneqq p - \epsilon \nabla_p E_I(p, a, b)$$

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

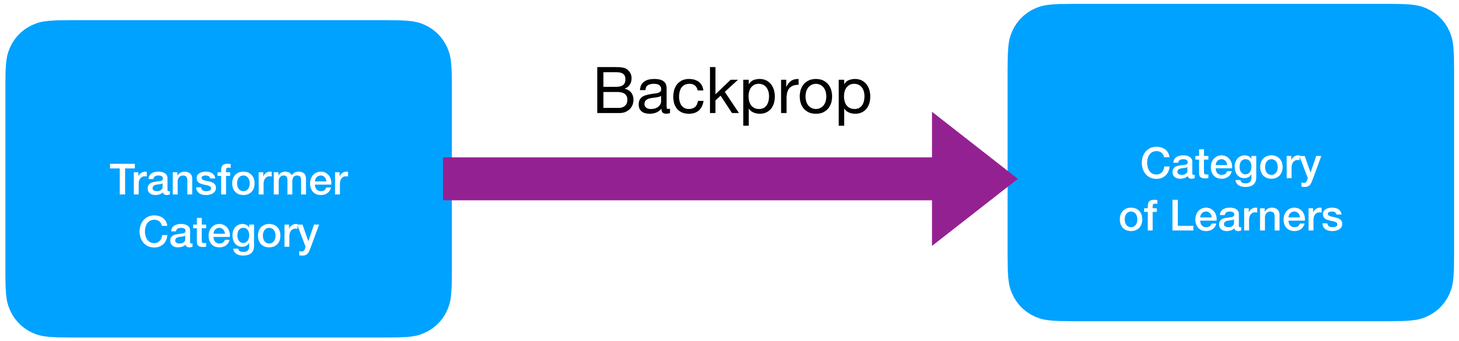




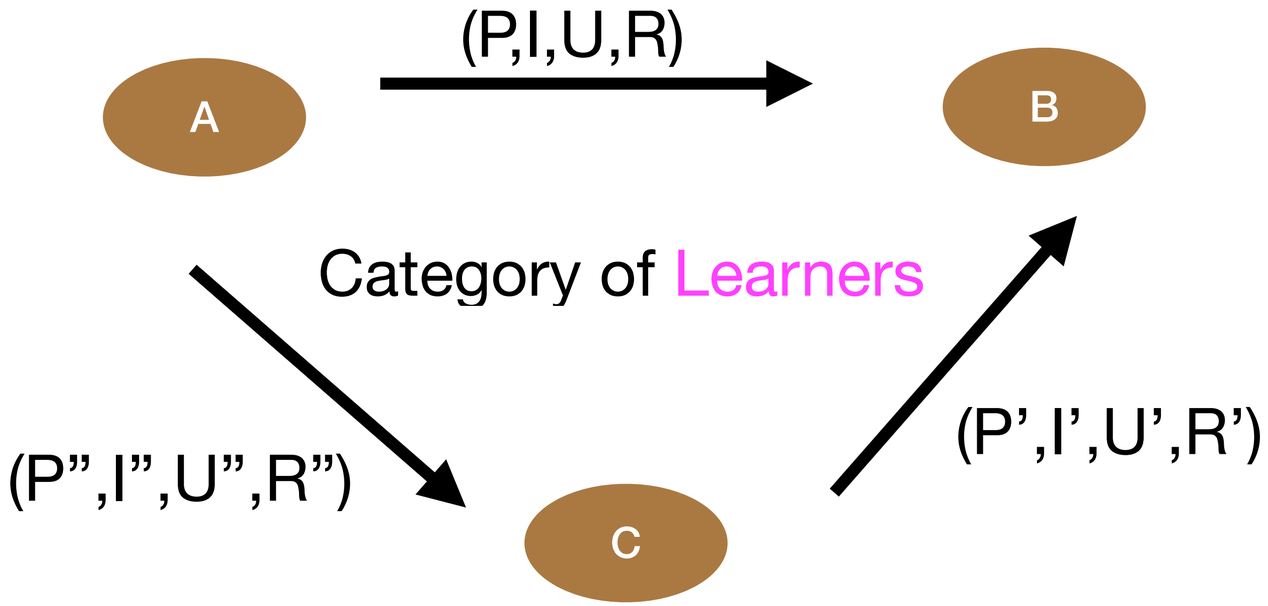

x
First-order oracle
{f(x), f'(x)}


x
Zeroth-order oracle
{f(x)}

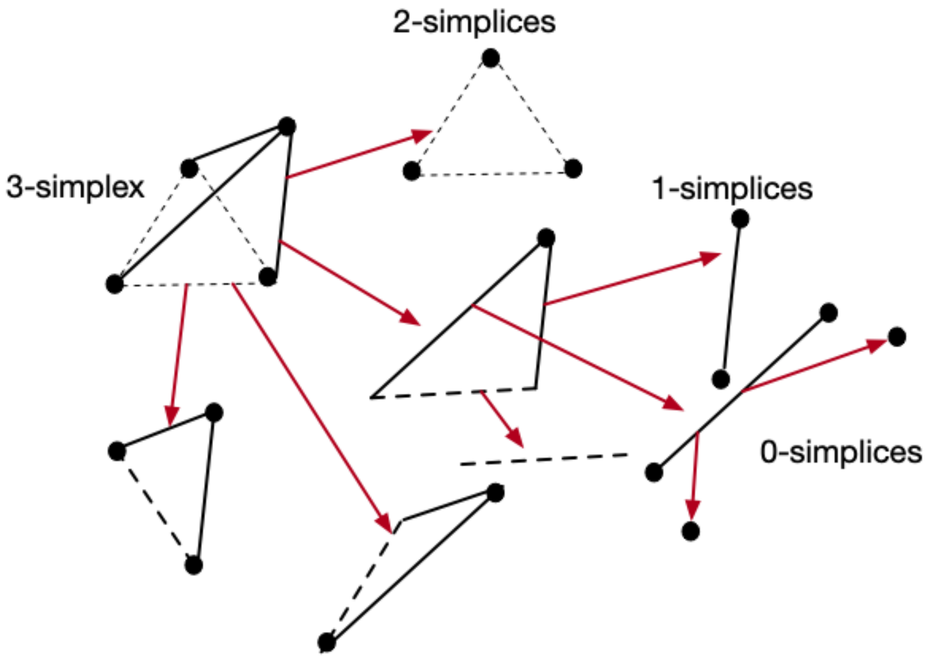
A Categorical Model of Attention

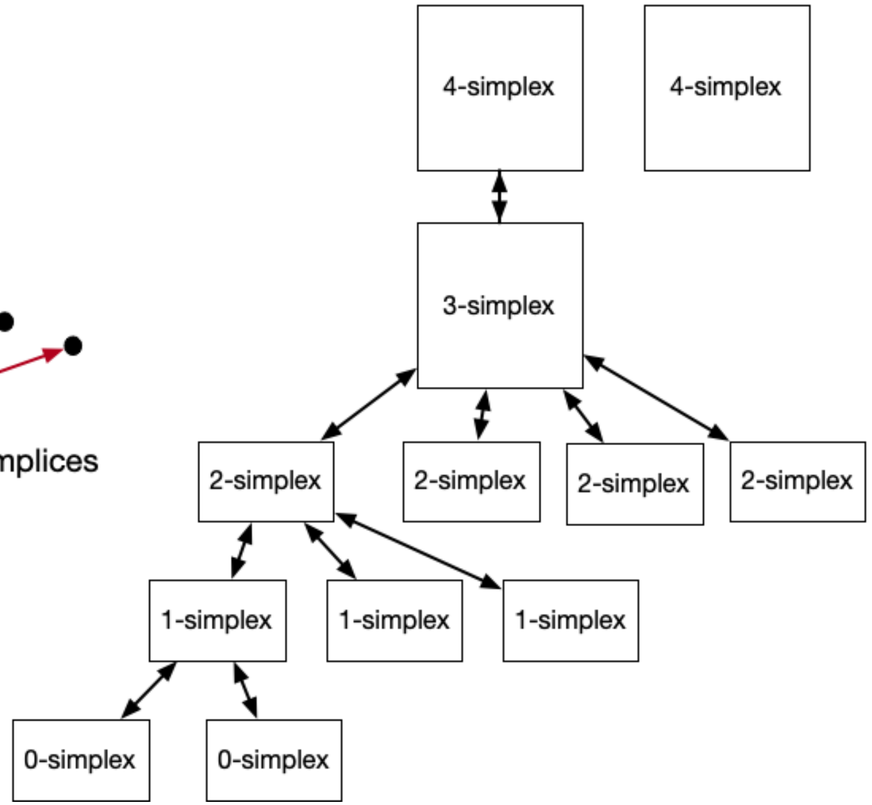


$$T^{h,m,r} \coloneqq \{f : \mathbb{R}^{d\times n} \to \mathbb{R}^{d\times n} \,|f(XP) = f(X)P\}$$

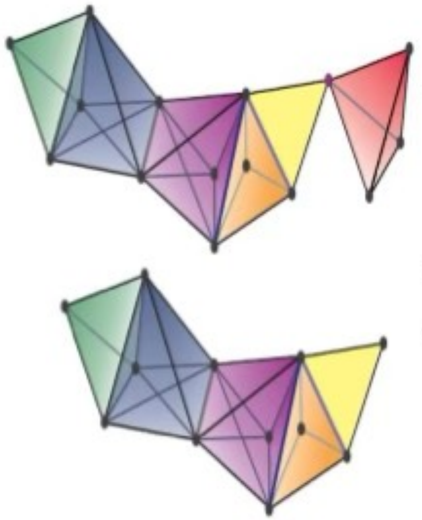


Simplicial Sets





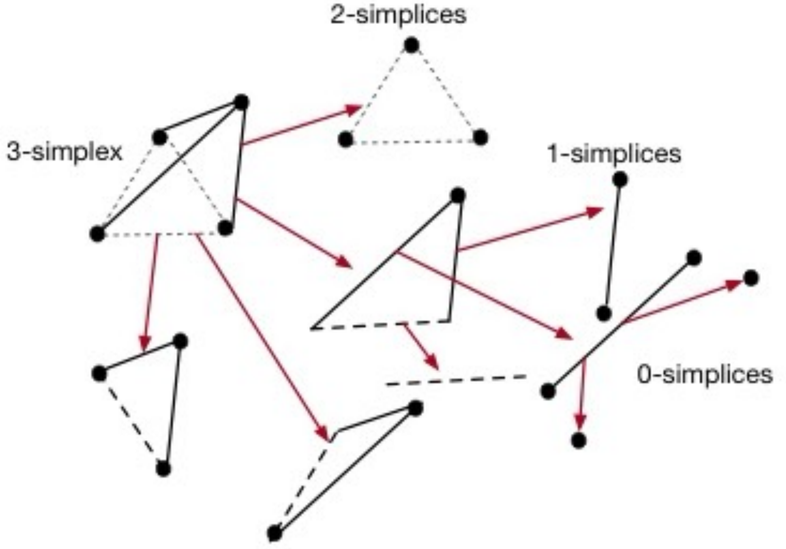
Simplicial framework for generative AI



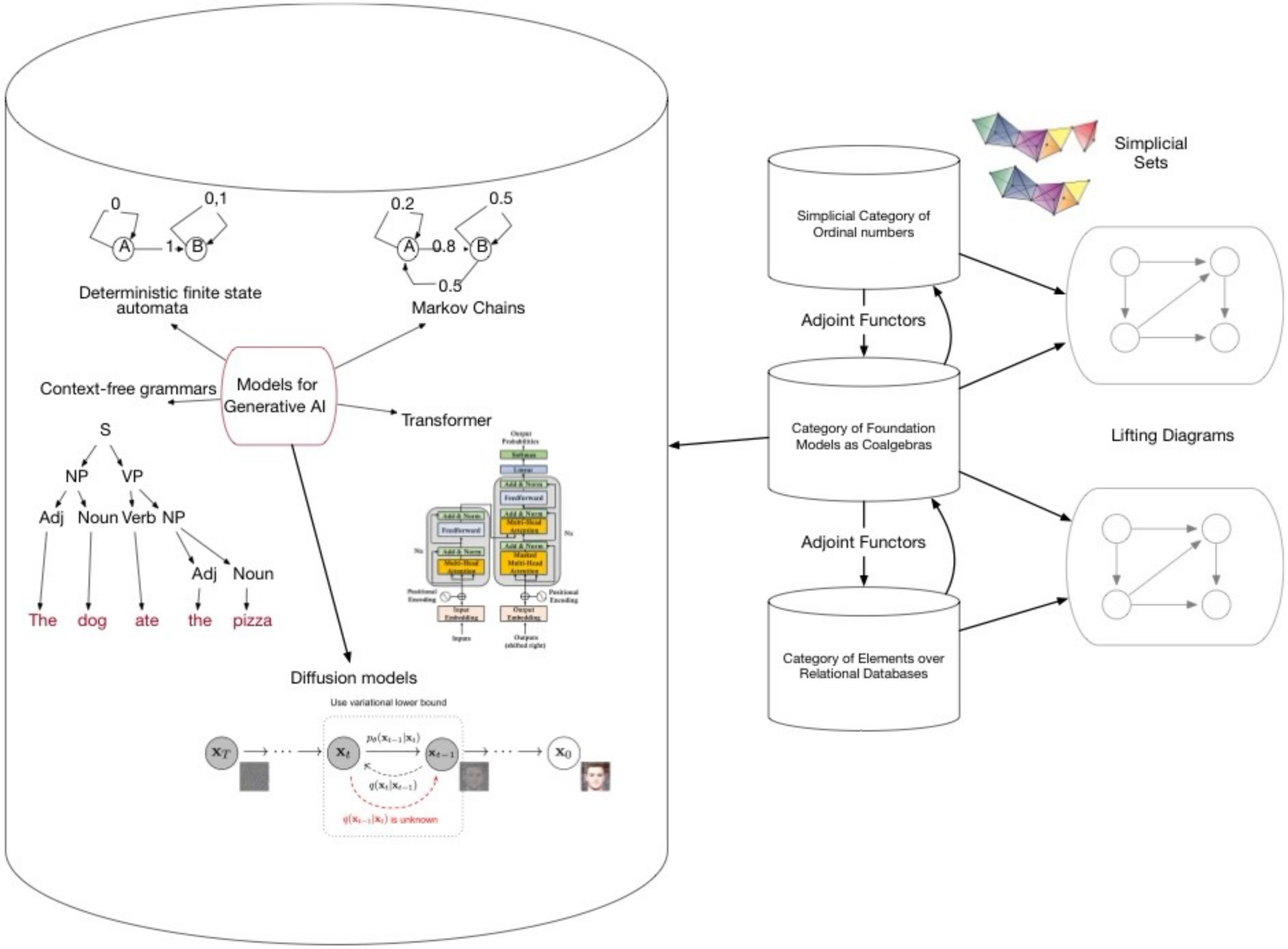
Simplicial learning is based on extension problems of inner and outer ``horns" of simplicial objects

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.

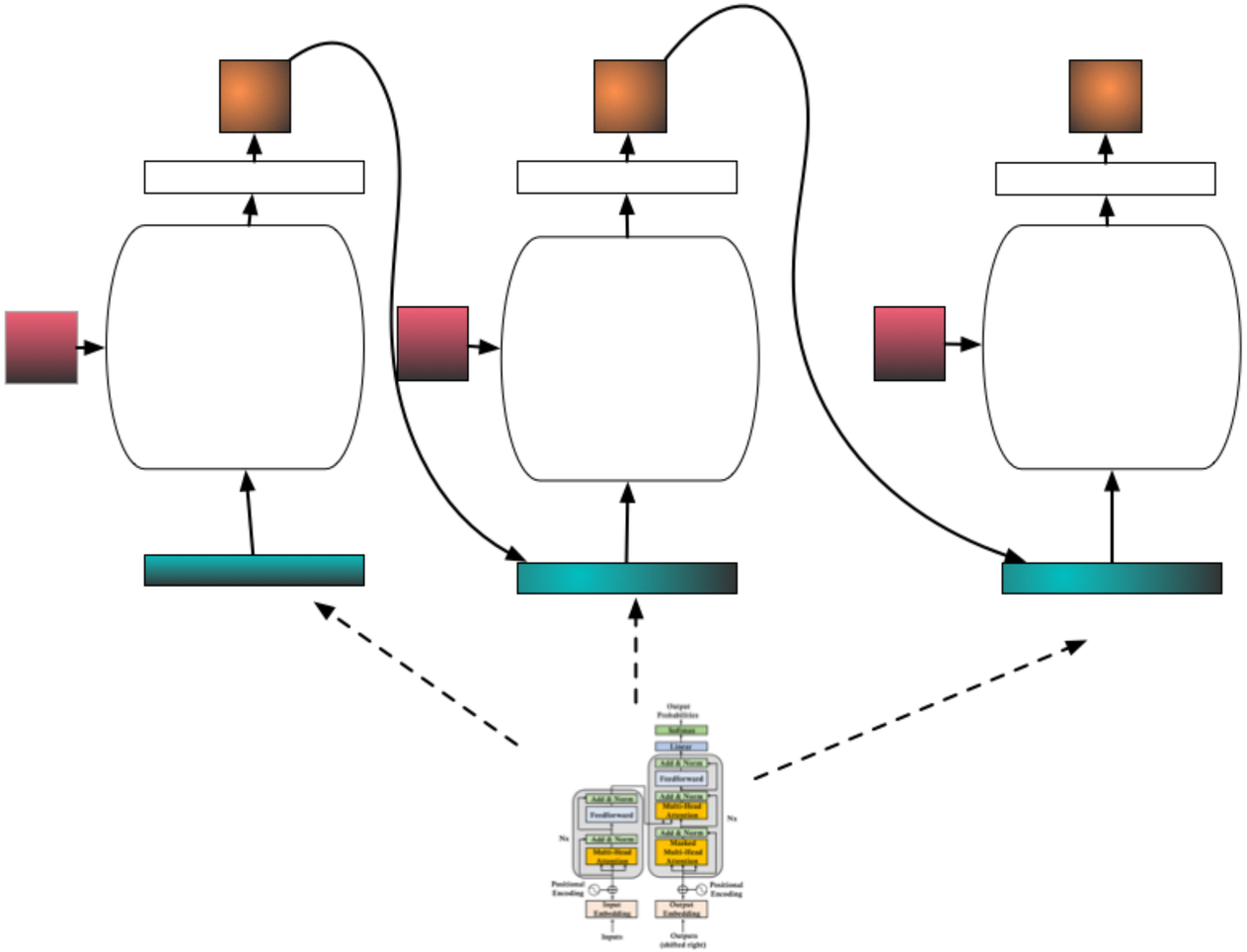


GAIA: Categorical Foundations of Generative AI

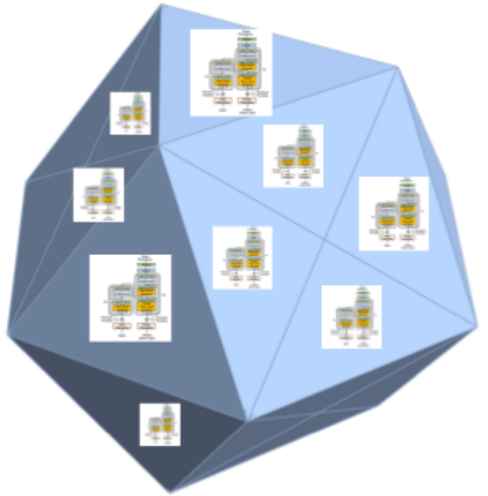


Simplicial Generative AI

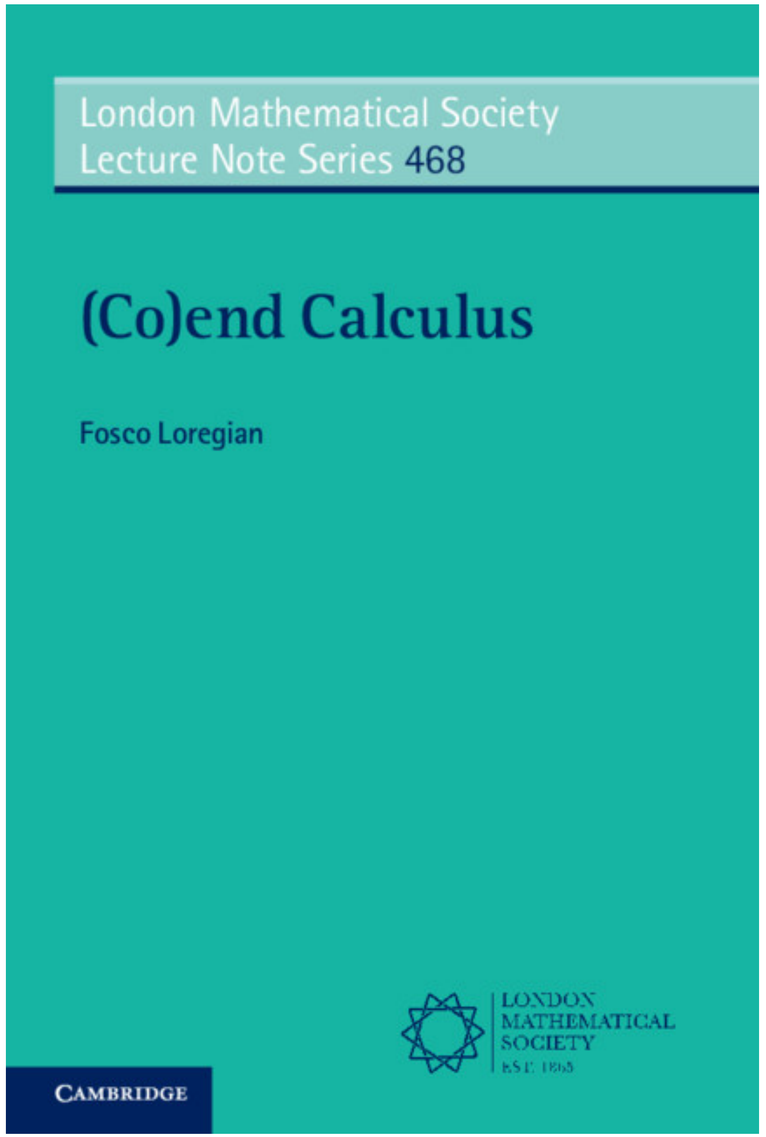
Sequential Generative AI Model



Simplicial Complex of Gen AI Models in GAIA

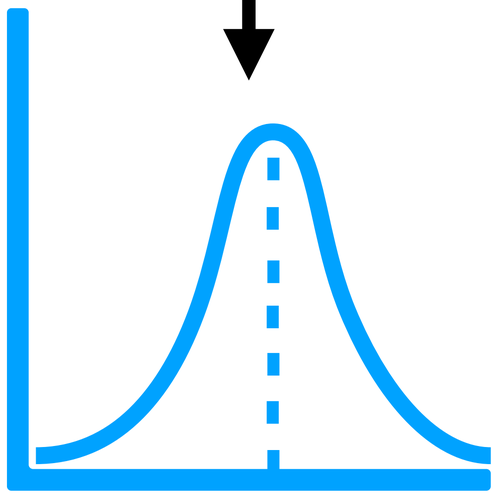


Calculus of (Co)Ends



$$\int_c F(c,c)$$

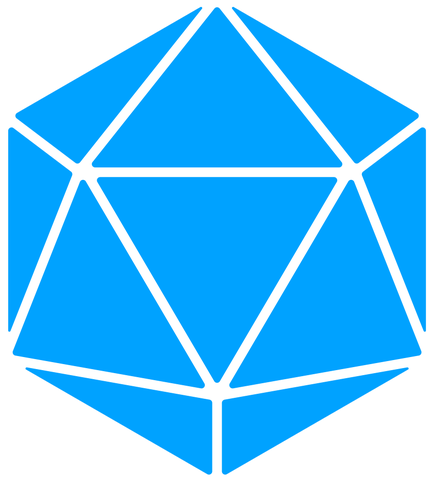
Ends



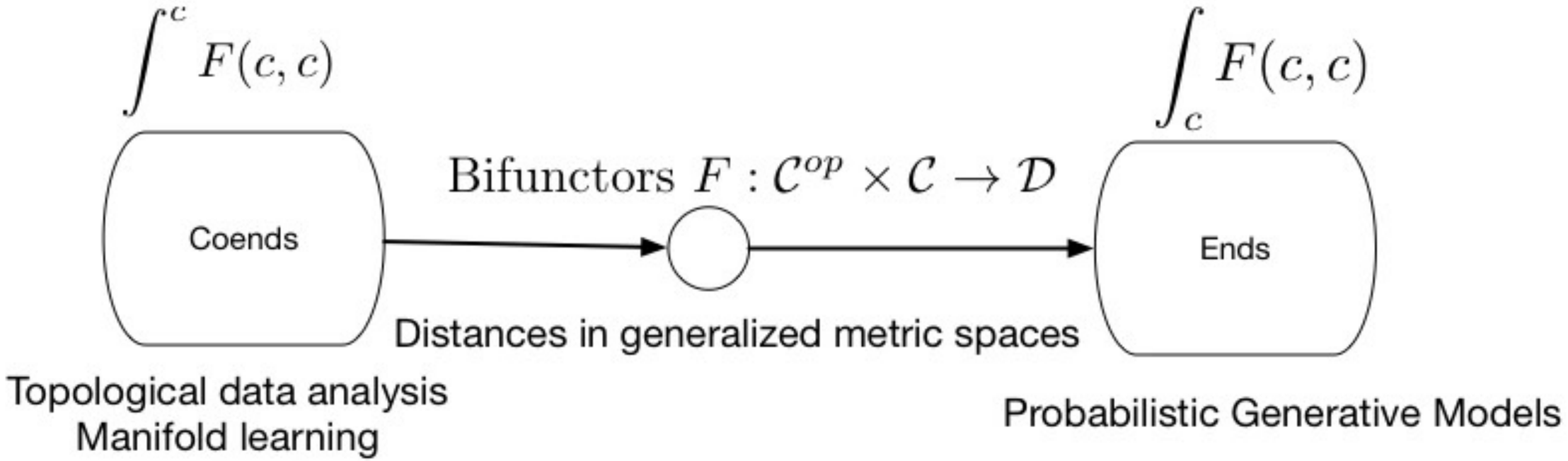
Probabilities

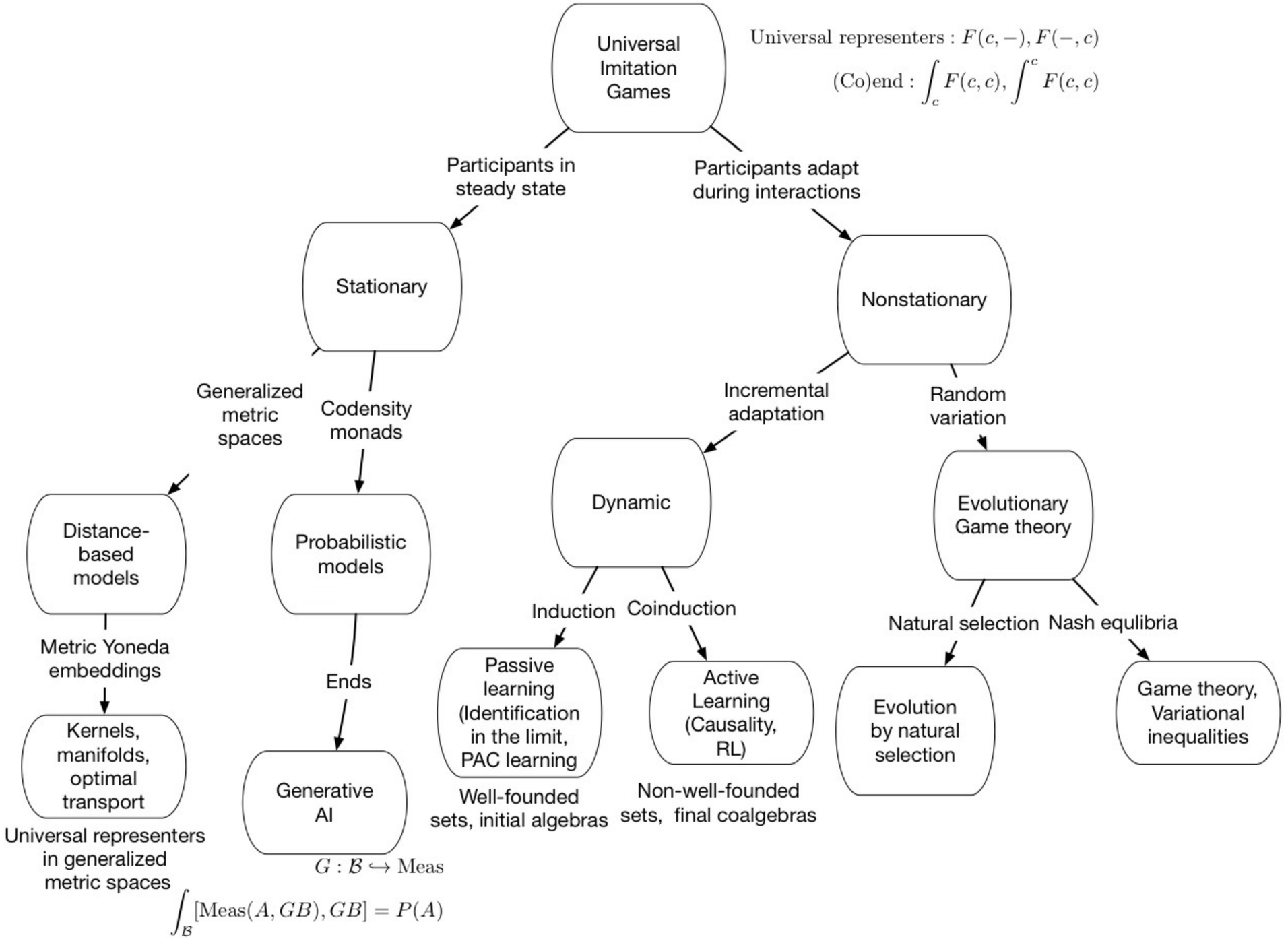
$$\int^c F(c,c)$$

Coends



Topology


$\int^{c} F(c,c)$
$\int_{c} F(c,c)$
Bifunctors $F : \mathcal{C}^{op} \times \mathcal{C} \to \mathcal{D}$
Coends
Ends
Distances in generalized metric spaces
Topological data analysis
Manifold learning
Probabilistic Generative Models


Universal Imitation Games
Universal representers : $F(c,-), F(-,c)$
(Co)end : $\int_c F(c,c), \int^c F(c,c)$
Participants in steady state
Participants adapt during interactions
Stationary
Nonstationary
Generalized metric spaces
Codensity monads
Incremental adaptation
Random variation
Distance-based models
Probabilistic models
Dynamic
Evolutionary Game theory
Metric Yoneda embeddings
Ends
Induction
Coinduction
Natural selection
Nash equlibria
Kernels, manifolds, optimal transport
Generative AI
Passive learning (Identification in the limit, PAC learning
Active Learning (Causality, RL)
Evolution by natural selection
Game theory, Variational inequalities
Universal representers in generalized metric spaces
Well-founded sets, initial algebras
Non-well-founded sets, final coalgebras
$G : \mathcal{B} \hookrightarrow \mathrm{Meas}$
$\int_{\mathcal{B}} [\mathrm{Meas}(A, GB), GB] = P(A)$

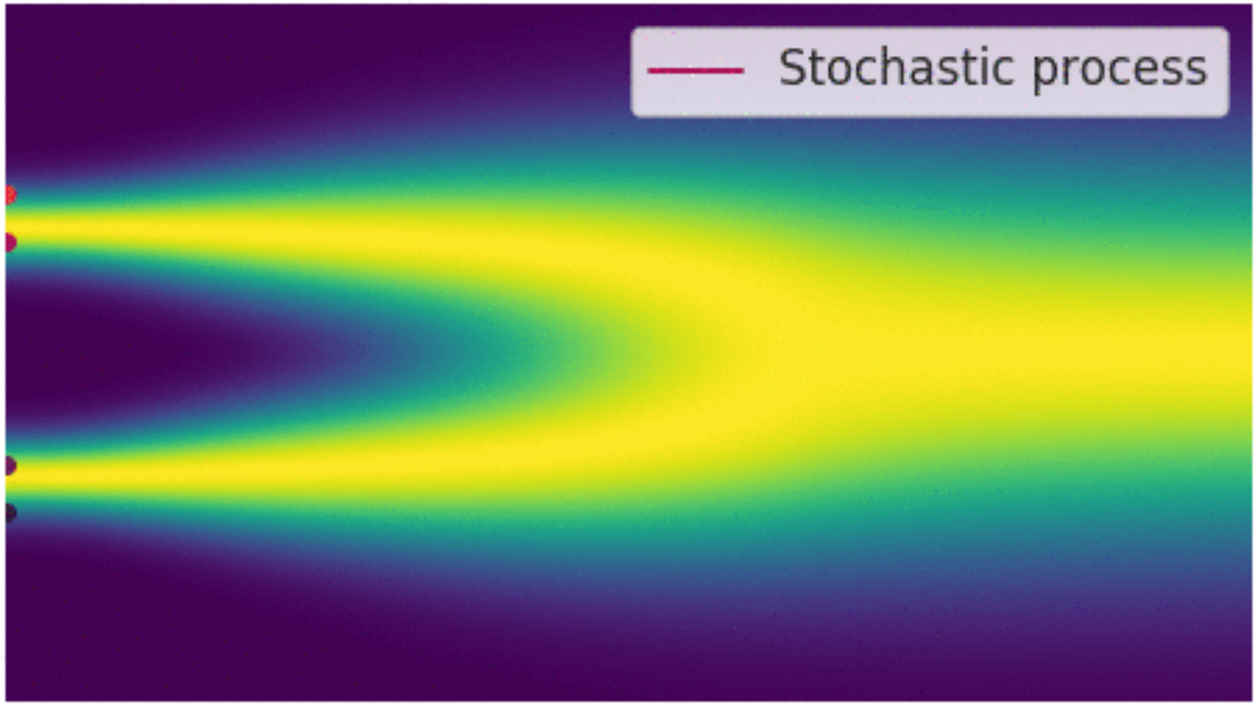
The “Geometric” Transformer Model

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

Intuition: Construct a simplicial set of of Transformers by composing sequences of length n

Embed them in a Kan complex

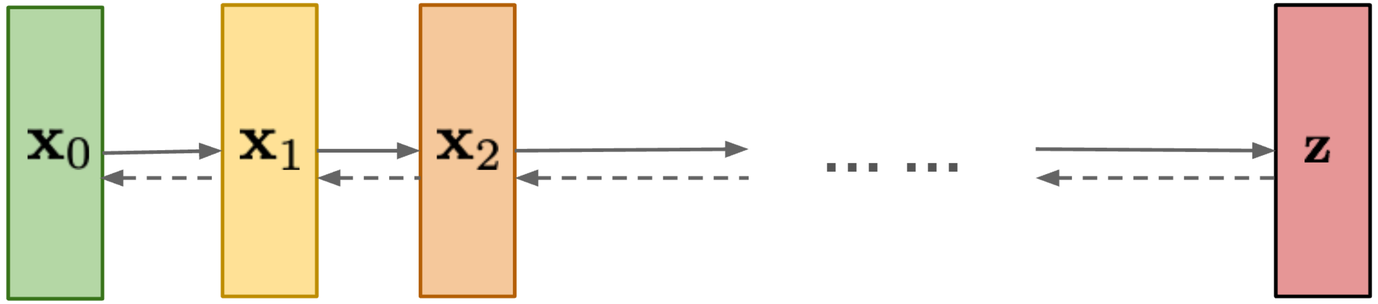
Diffusion Process and Kan Complexe



https://yang-song.net/blog/2021/score/

Generative AI and Kan Complexes

Diffusion models:
Gradually add Gaussian noise and then reverse



Every morphism

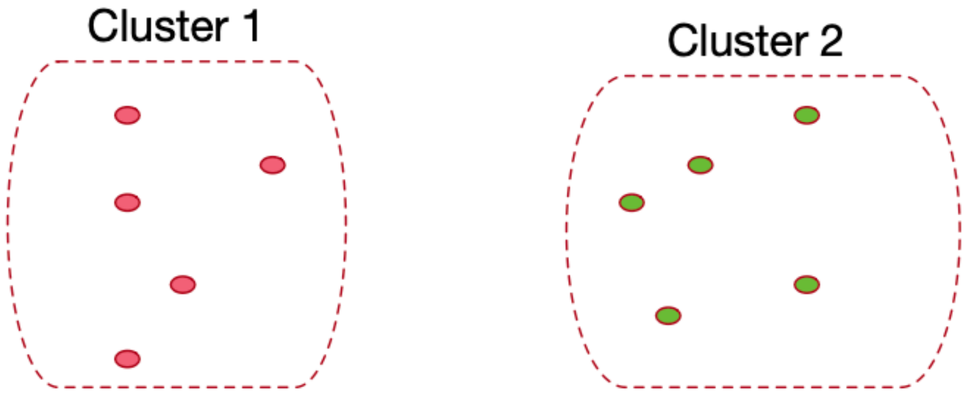
invertible!

Figure Source: https://lilianweng.github.io/posts/2021-07-11-diffusion-models/

Categorical Machine Learning

Clustering as a Functor

- *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. In terms of Figure 9, 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 in Figure 9 were decreased, and the distances between points in different clusters were increased, the clustering should not change either.



An Impossibility Theorem for Clustering

Jon Kleinberg
Department of Computer Science
Cornell University
Ithaca NY 14853

Abstract

Although the study of *clustering* is centered around an intuitively compelling goal, it has been very difficult to develop a unified framework for reasoning about it at a technical level, and profoundly diverse approaches to clustering abound in the research community. Here we suggest a formal perspective on the difficulty in finding such a unification, in the form of an *impossibility theorem*: for a set of three simple properties, we show that there is no clustering function satisfying all three. Relaxations of these properties expose some of the interesting (and unavoidable) trade-offs at work in well-studied clustering techniques such as single-linkage, sum-of-pairs, k-means, and k-median.

Journal of Machine Learning Research 11 (2010) 1425-1470 Submitted 4/09; Revised 12/09; Published 4/10

Characterization, Stability and Convergence of Hierarchical Clustering Methods

Gunnar Carlsson GUNNAR@MATH.STANFORD.EDU
Facundo Mémoli* MEMOLI@MATH.STANFORD.EDU
Department of Mathematics
Stanford University
Stanford, CA 94305

Editor: Ulrike von Luxburg

Abstract

We study hierarchical clustering schemes under an axiomatic view. We show that within this framework, one can prove a theorem analogous to one of Kleinberg (2002), in which one obtains an existence and uniqueness theorem instead of a non-existence result. We explore further properties of this unique scheme: stability and convergence are established. We represent dendrograms as ultrametric spaces and use tools from metric geometry, namely the Gromov-Hausdorff distance, to quantify the degree to which perturbations in the input metric space affect the result of hierarchical methods.

Keywords: clustering, hierarchical clustering, stability of clustering, Gromov-Hausdorff distance

UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction

Uniform Manifold Approximation and Projection (UMAP) is a dimension reduction technique that can be used for visualisation similarly to t-SNE, but also for general non-linear dimension reduction. The algorithm is founded on three assumptions about the data

1. The data is uniformly distributed on Riemannian manifold;
2. The Riemannian metric is locally constant (or can be approximated as such);
3. The manifold is locally connected.

From these assumptions it is possible to model the manifold with a fuzzy topological structure. The embedding is found by searching for a low dimensional projection of the data that has the closest possible equivalent fuzzy topological structure.

The details for the underlying mathematics can be found in our paper on ArXiv:

McInnes, L, Healy, J, *UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction*, ArXiv e-prints 1802.03426, 2018

You can find the software on github.

Installation

UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction

Leland McInnes
Tutte Institute for Mathematics and Computing
leland.mcinnes@gmail.com

John Healy
Tutte Institute for Mathematics and Computing
jchealy@gmail.com

James Melville
jlmelville@gmail.com

September 21, 2020

Abstract

UMAP (Uniform Manifold Approximation and Projection) is a novel manifold learning technique for dimension reduction. UMAP is constructed from a theoretical framework based in Riemannian geometry and algebraic topology. The result is a practical scalable algorithm that is applicable to real world data. The UMAP algorithm is competitive with t-SNE for visualization quality, and arguably preserves more of the global structure with superior run time performance. Furthermore, UMAP has no computational restrictions on embedding dimension, making it viable as a general purpose dimension reduction technique for machine learning.

1 Introduction

Dimension reduction plays an important role in data science, being a fundamental technique in both visualisation and as pre-processing for machine

Definition 1. *The category* $\mathbf{\Delta}$ *has as objects the finite order sets* $[n] = \{1, \dots, n\}$*, with morphims given by (non-strictly) order-preserving maps.*

Following standard category theoretic notation, $\mathbf{\Delta}^{\mathrm{op}}$ denotes the category with the same objects as $\mathbf{\Delta}$ and morphisms given by the morphisms of $\mathbf{\Delta}$ with the direction (domain and codomain) reversed.

Definition 2. *A* simplicial set *is a functor from* $\mathbf{\Delta}^{op}$ *to* **Sets**, *the category of sets; that is, a contravariant functor from* $\mathbf{\Delta}$ *to* **Sets**.

Given a simplicial set $X : \mathbf{\Delta}^{\mathrm{op}} \to \mathbf{Sets}$, it is common to denote the set $X([n])$ as X_n and refer to the elements of the set as the n-simplices of X. The simplest possible examples of simplicial sets are the *standard simplices* Δ^n, defined as the representable functors $\hom_{\mathbf{\Delta}}(\cdot, [n])$. It follows from the Yoneda lemma that there is a natural correspondence between n-simplices of X and morphisms $\Delta^n \to X$ in the category of simplicial sets, and it is often helpful to think in these terms. Thus for each $x \in X_n$ we have a corresponding morphism $x : \Delta^n \to X$. By the density theorem and employing a minor abuse of notation we then have

$$\operatorname*{colim}_{x \in X_n} \Delta^n \cong X$$

There is a standard covariant functor $|\cdot| : \mathbf{\Delta} \to \mathbf{Top}$ mapping from the category $\mathbf{\Delta}$ to the category of topological spaces that sends $[n]$ to the standard n-simplex $|\Delta^n| \subset \mathbb{R}^{n+1}$ defined as

$$|\Delta^n| \triangleq \left\{ (t_0, \dots, t_n) \in \mathbb{R}^{n+1} \mid \sum_{i=0}^{n} t_i = 1, t_i \geq 0 \right\}$$

with the standard subspace topology. If $X : \mathbf{\Delta}^{\mathrm{op}} \to \mathbf{Sets}$ is a simplicial set then we can construct the realization of X (denoted $|X|$) as the colimit

$$|X| = \operatorname*{colim}_{x \in X_n} |\Delta^n|$$

Definition 7. *Define the functor* $\mathsf{FinReal} : \mathbf{Fin\text{-}sFuzz} \to \mathbf{FinEPMet}$ *by setting*

$$\mathsf{FinReal}(\Delta^n_{<a}) \triangleq (\{x_1, x_2, \ldots, x_n\}, d_a),$$

where

$$d_a(x_i, x_j) = \begin{cases} -\log(a) & \text{if } i \neq j, \\ 0 & \text{otherwise} \end{cases}.$$

and then defining

$$\mathsf{FinReal}(X) \triangleq \operatorname*{colim}_{\Delta^n_{<a} \to X} \mathsf{FinReal}(\Delta^n_{<a}).$$

[McInnes et al., 2020]

Definition 8. *Define the functor* FinSing : **FinEPMet** $\to$ **Fin-sFuzz** *by*

$$\mathsf{FinSing}(Y) : ([n], [0, a)) \mapsto \hom_{\mathbf{FinEPMet}}(\mathsf{FinReal}(\Delta^n_{<a}), Y).$$

We then have the following theorem.

Theorem 1. *The functors* FinReal : **Fin-sFuzz** $\to$ **FinEPMet** *and* FinSing : **FinEPMet** $\to$ **Fin-sFuzz** *form an adjunction with* FinReal *the left adjoint and* FinSing *the right adjoint.*

[McInnes et al., 2020]

Algorithm 1 UMAP algorithm

function UMAP(X, n, d, min-dist, n-epochs)

 # Construct the relevant weighted graph

for all $x \in X$ **do**

fs-set$[x] \leftarrow$ LocalFuzzySimplicialSet(X, x, n)

top-rep $\leftarrow \bigcup_{x \in X}$ fs-set$[x]$ *# We recommend the probabilistic t-conorm*

 # Perform optimization of the graph layout

$Y \leftarrow$ SpectralEmbedding(top-rep, d)

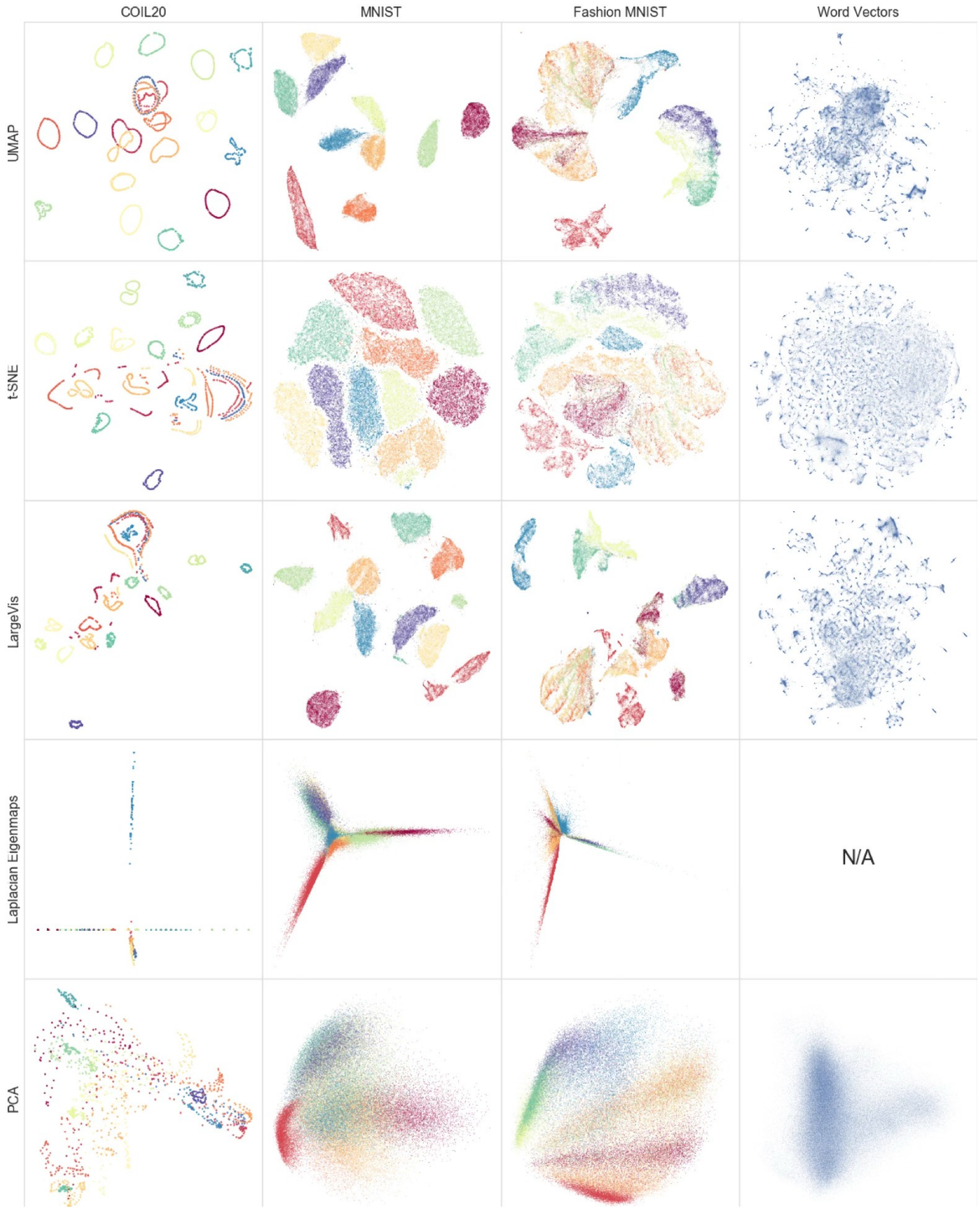
$Y \leftarrow$ OptimizeEmbedding(top-rep, Y, min-dist, n-epochs)

return Y

[McInnes et al., 2020]

UMAP

PCA


COIL20
MNIST
Fashion MNIST
Word Vectors
UMAP
t-SNE
LargeVis
Laplacian Eigenmaps
PCA
N/A

https://doi.org/10.1038/s41586-019-0969-x

The single-cell transcriptional landscape of mammalian organogenesis

Junyue Cao[1,2,10], Malte Spielmann[1,10], Xiaojie Qiu[1,2], Xingfan Huang[1,3], Daniel M. Ibrahim[4,5], Andrew J. Hill[1], Fan Zhang[6], Stefan Mundlos[4,5], Lena Christiansen[6], Frank J. Steemers[6], Cole Trapnell[1,7,8]* & Jay Shendure[1,7,8,9]*

Mammalian organogenesis is a remarkable process. Within a short timeframe, the cells of the three germ layers transform into an embryo that includes most of the major internal and external organs. Here we investigate the transcriptional dynamics of mouse organogenesis at single-cell resolution. Using single-cell combinatorial indexing, we profiled the transcriptomes of around 2 million cells derived from 61 embryos staged between 9.5 and 13.5 days of gestation, in a single experiment. The resulting 'mouse organogenesis cell atlas' (MOCA) provides a global view of developmental processes during this critical window. We use Monocle 3 to identify hundreds of cell types and 56 trajectories, many of which are detected only because of the depth of cellular coverage, and collectively define thousands of corresponding marker genes. We explore the dynamics of gene expression within cell types and trajectories over time, including focused analyses of the apical ectodermal ridge, limb mesenchyme and skeletal muscle.

Most studies of mammalian organogenesis rely on model organisms, and, in particular, the mouse. Mice develop quickly, with just 21 days between fertilization and birth. The implantation of the blastocyst on embryonic day (E) 4.0 is followed by gastrulation and the formation of germ layers on E6.5–E7.5[1,2]. At the early-somite stages, the embryo transits from gastrulation to early organogenesis, forming the neural plate and heart tube (E8.0–E8.5). In the ensuing days (E9.5–E13.5), the embryo expands from hundreds-of-thousands to over ten-million cells, and concurrently develops nearly all major organ systems. Unsurprisingly, these four days have been intensively studied. Indeed, most genes that underlie major developmental defects can be studied in this window[3,4].

The transcriptional profiling of single cells (scRNA-seq) represents a promising strategy for obtaining a global view of developmental processes[5–7]. For example, scRNA-seq recently revealed a large degree of heterogeneity in neurons and myocardiocytes during mouse development[8,9]. However, although two scRNA-seq atlases of the mouse were recently released[10,11], they are mostly restricted to adult organs and do not attempt to characterize the emergence and dynamics of cell types during development.

Single-cell RNA-seq of two million cells

Single-cell combinatorial indexing is a methodological framework involving split-pool barcoding of cells or nuclei[12–19]. We previously developed single-cell combinatorial-indexing RNA-sequencing analysis (sci-RNA-seq) and applied it to generate 50-fold shotgun coverage of the cellular content of L2-stage *Caenorhabditis elegans*[17]. A conceptually identical method was recently termed SPLiT-seq[20]. To increase the throughput, we explored more than 1,000 experimental conditions (Extended Data Fig. 1a, b, Methods). The major improvements of the resulting method, sci-RNA-seq3, include: (i) nuclei are extracted directly from fresh tissues without enzymatic treatment, then fixed and stored; (ii) for the third level of indexing[17], we switched from Tn5 tagmentation to hairpin ligation; (iii) individual enzymatic reactions were optimized; and (iv) fluorescence-activated cell sorting was replaced by dilution, and sonication and filtration steps were added to minimize aggregation. Even without automation, sci-RNA-seq3 library preparation can be completed through the intensive effort of a single researcher in one week at a cost of less than $0.01 per cell.

We collected 61 C57BL/6 mouse embryos at E9.5, E10.5, E11.5, E12.5 or E13.5, and snap-froze them in liquid nitrogen. Nuclei from each embryo were isolated and deposited in individual wells in four 96-well plates, such that the first index identified the originating embryo of a given cell. As a control, we spiked a mixture of human HEK-293T and mouse NIH/3T3 nuclei into two wells. The resulting sci-RNA-seq3 library was sequenced in a single Illumina NovaSeq run, yielding 11 billion reads (Fig. 1a, Extended Data Fig. 1c, d).

From one experiment, we recovered 2,058,652 cells from mouse embryos and 13,359 cells from HEK-293T or NIH/3T3 cells (UMI (unique molecular identifier) count $\geq$ 200). Transcriptomes from human or mouse control wells were overwhelmingly species-coherent (3% collisions), with performance similar to previous experiments[17] (Extended Data Fig. 1e–i). A limitation is that only around 7% of cells entering the experiment were ultimately profiled, with losses largely consequent on filtration steps intended to remove aggregates of nuclei.

We profiled a median of 35,272 cells per embryo (Fig. 1b, Extended Data Fig. 1j). Despite shallow sequencing (about 5,000 raw reads per cell; 46% duplicate rate), we recovered a median of 671 UMIs (519 genes) per cell (Extended Data Fig. 1k). The 3.7-fold-deeper sequencing of a subset of wells nearly doubled the complexity (to a median of 1,142 UMIs per cell; 87% duplicate rate). Given that we are profiling RNA in nuclei, 59% of UMIs per cell strand specifically mapped to introns and 25% mapped to exons. The profiles may therefore primarily reflect nascent transcription, temporally offset, but also predictive[21] of the cellular transcriptome. Later-stage embryos exhibited somewhat reduced UMI counts, possibly reflecting decreasing nuclear mRNA content (Extended Data Fig. 1l). We used Scrublet[22] to detect 4.3% likely doublet cells, corresponding to a doublet estimate of 10.3% including both within-cluster and between-cluster doublets (Extended Data Fig. 1m, n).

[1]Department of Genome Sciences, University of Washington, Seattle, WA, USA. [2]Molecular and Cellular Biology Program, University of Washington, Seattle, WA, USA. [3]Department of Computer Science, University of Washington, Seattle, WA, USA. [4]Max Planck Institute for Molecular Genetics, RG Development & Disease, Berlin, Germany. [5]Institute for Medical and Human Genetics, Charité Universitätsmedizin Berlin, Berlin, Germany. [6]Illumina, San Diego, CA, USA. [7]Brotman Baty Institute for Precision Medicine, Seattle, WA, USA. [8]Allen Discovery Center for Cell Lineage Tracing, Seattle, WA, USA. [9]Howard Hughes Medical Institute, Seattle, WA, USA. [10]These authors contributed equally: Junyue Cao, Malte Spielmann. *e-mail: coletrap@uw.edu; shendure@uw.edu

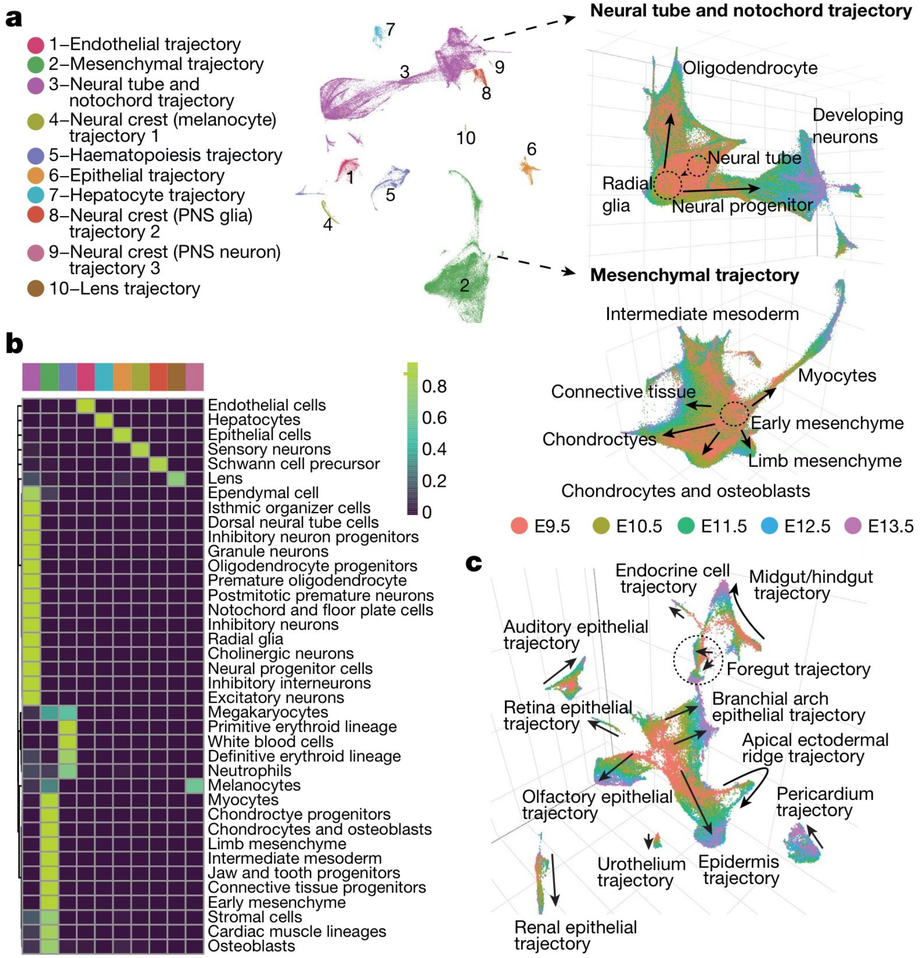
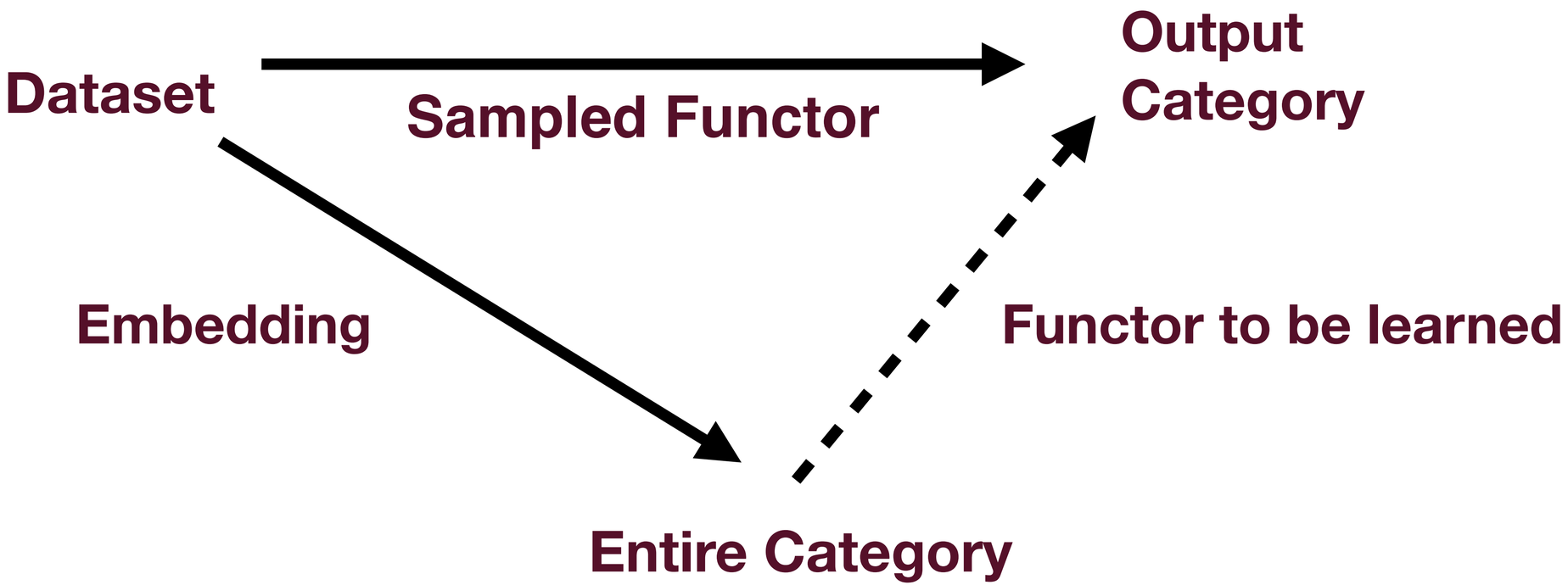


Fig. 4 | Characterization of ten major developmental trajectories present during mouse organogenesis. a, UMAP 3D visualization of our overall dataset. Left, views from one direction; bottom: zoomed view of neural tube–notochord (top) and mesenchymal (bottom) trajectories, coloured by development stage. **b**, Heat map showing the proportion of cells from each of the 38 major cell types (rows) assigned to each of the 10 major trajectories (columns, colour key in **a**, left). **c**, UMAP 3D visualization of epithelial subtrajectories coloured by development stage (colour key in **a**, right).

simplicial set representations. UMAP then optimizes the lower-dimension embedding, minimizing the cross-entropy between the low-dimensional representation and the high-dimensional one.

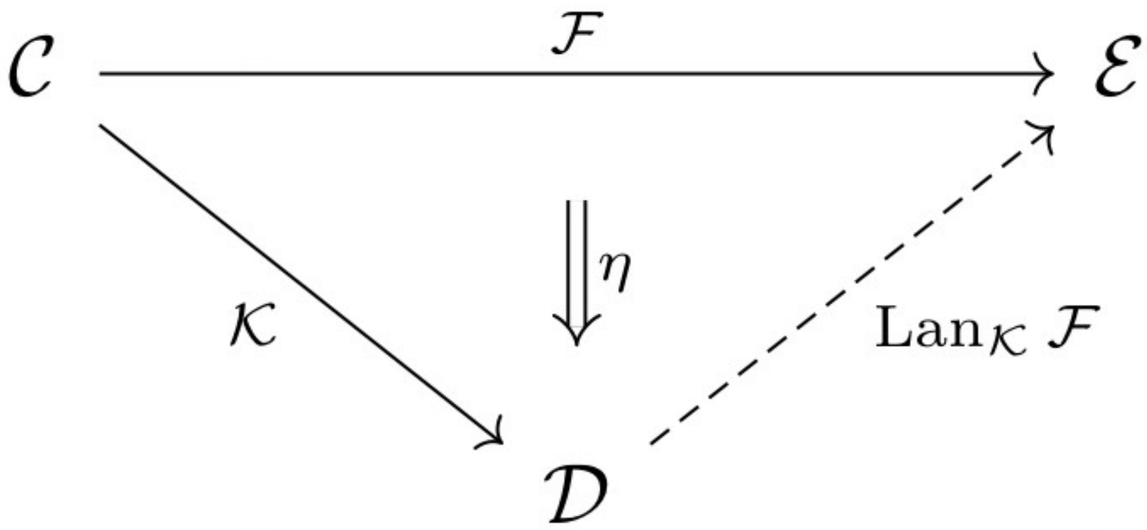
The computational efficiency of UMAP markedly accelerated the analysis of the mouse embryo data. We found that UMAP finished processing the two-million-cell dataset in around 3 CPU hours whereas *t*-SNE took more than 64 CPU hours. A few implementation details lead to the effectiveness of UMAP. Two major steps are involved in both the UMAP and *t*-SNE algorithms: first, the preprocessing step before UMAP is similar to Monocle 2. In brief, genes expressed in fewer than 10 cells (or fewer than 5 cells in datasets with fewer than 1,000 cells) were filtered out. The digital gene-count matrix was first normalized by cell-specific size factor

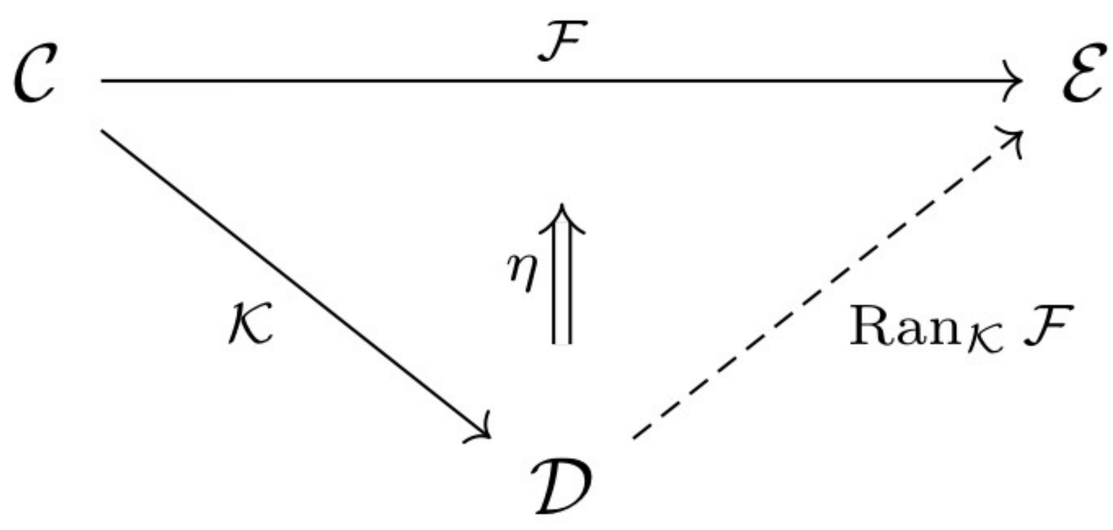
Learning Functors



Find the universal property!

Kan Extensions of Functors





Every concept in category theory can be expressed as a Kan extension!

Dynamical Systems and Universal Coalgebras

Universal Reinforcement Learning

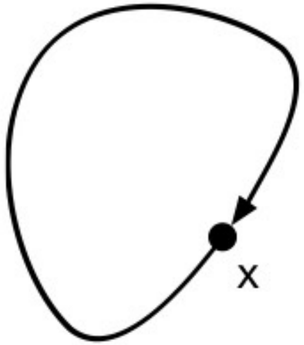
Lecture Notes

NON-WELL-FOUNDED SETS

Peter Aczel

Foreword by Jon Barwise

CSLI
CENTER FOR THE STUDY
OF LANGUAGE
AND INFORMATION



$x = \{x\}$

$x \longrightarrow F(x)$

From Induction to Coinduction

- **Machine learning has traditionally been modeled as induction**
- **Identification in the limit: Gold, Solomonoff**
- **PAC Learning: Valiant, Vapnik**
- **Algorithmic Information Theory: Chaitin, Kolmogorov**
- **Occam’s Razor, Minimum Description Length**

Coinduction: A New Paradigm for ML

- Generative AI is all about modeling infinite data streams
  - Automata, Grammars, Markov processes, LLMs, diffusion models
- Infinite data streams define non-well-founded sets
- Final coalgebras generalize (greatest) fixed points
- Reinforcement learning is an example of coinduction in a coalgebra
- Causal inference is also usefully modeled in coalgebras

The Method of Coalgebra: exercises in coinduction

Jan Rutten

February 2019
261 pages
ISBN 978-90-6196-568-8

Publisher: CWI, Amsterdam,
The Netherlands

ELSEVIER

Available at
www.ComputerScienceWeb.com
POWERED BY SCIENCE @DIRECT®

Theoretical Computer Science 308 (2003) 1–53

Theoretical
Computer Science

www.elsevier.com/locate/tcs

Fundamental Study

Behavioural differential equations: a coinductive calculus of streams, automata, and power series☆

J.J.M.M. Rutten

CWI, P.O. Box 94079, 1090 GB Amsterdam, The Netherlands

Received 25 October 2000; received in revised form 11 November 2002; accepted 14 November 2002
Communicated by D. Sannella

Abstract

We present a theory of streams (infinite sequences), automata and languages, and formal power series, in terms of the notions of homomorphism and bisimulation, which are the cornerstones of the theory of (universal) coalgebra. This coalgebraic perspective leads to a unified theory, in which the observation that each of the aforementioned sets carries a so-called *final* automaton structure, plays a central role. Finality forms the basis for both definitions and proofs by coinduction, the coalgebraic counterpart of induction. Coinductive definitions take the shape of what we have called behavioural differential equations, after Brzozowski's notion of input derivative. A calculus is developed for coinductive reasoning about all of the afore mentioned structures, closely resembling calculus from classical analysis.

Conductive Inference

- Based on non-well-founded sets
- Uses the category-theoretic framework of universal coalgebras
- Coinduction generalizes (greatest) fixed point analysis
- Reinforcement learning: metric coinduction in stochastic coalgebras

ELSEVIER

Theoretical Computer Science 249 (2000) 3–80

Theoretical Computer Science

www.elsevier.com/locate/tcs

Fundamental Study

Universal coalgebra: a theory of systems

J.J.M.M. Rutten

CWI, P.O. Box 94079, 1090 GB Amsterdam, Netherlands

Communicated by M.W. Mislove

Abstract

In the semantics of programming, finite data types such as finite lists, have traditionally been modelled by initial algebras. Later final *coalgebras* were used in order to deal with *infinite* data types. Coalgebras, which are the dual of algebras, turned out to be suited, moreover, as models for certain types of automata and more generally, for (transition and dynamical) *systems*. An important property of initial algebras is that they satisfy the familiar principle of induction. Such a principle was missing for coalgebras until the work of Aczel (Non-Well-Founded sets, CSLI Leethre Notes, Vol. 14, center for the study of Languages and information, Stanford, 1988) on a theory of non-wellfounded sets, in which he introduced a proof principle nowadays called *coinduction*. It was formulated in terms of *bisimulation*, a notion originally stemming from the world of concurrent programming languages. Using the notion of *coalgebra homomorphism*, the definition of bisimulation on coalgebras can be shown to be formally dual to that of congruence on algebras. Thus, the three basic notions of universal algebra: algebra, homomorphism of algebras, and congruence, turn out to correspond to coalgebra, homomorphism of coalgebras, and bisimulation, respectively. In this paper, the latter are taken as the basic ingredients of a theory called *universal coalgebra*. Some standard results from universal algebra are reformulated (using the aforementioned correspondence) and proved for a large class of coalgebras, leading to a series of results on, e.g., the lattices of subcoalgebras and bisimulations, simple coalgebras and coinduction, and a covariety theorem for coalgebras similar to Birkhoff's variety theorem.

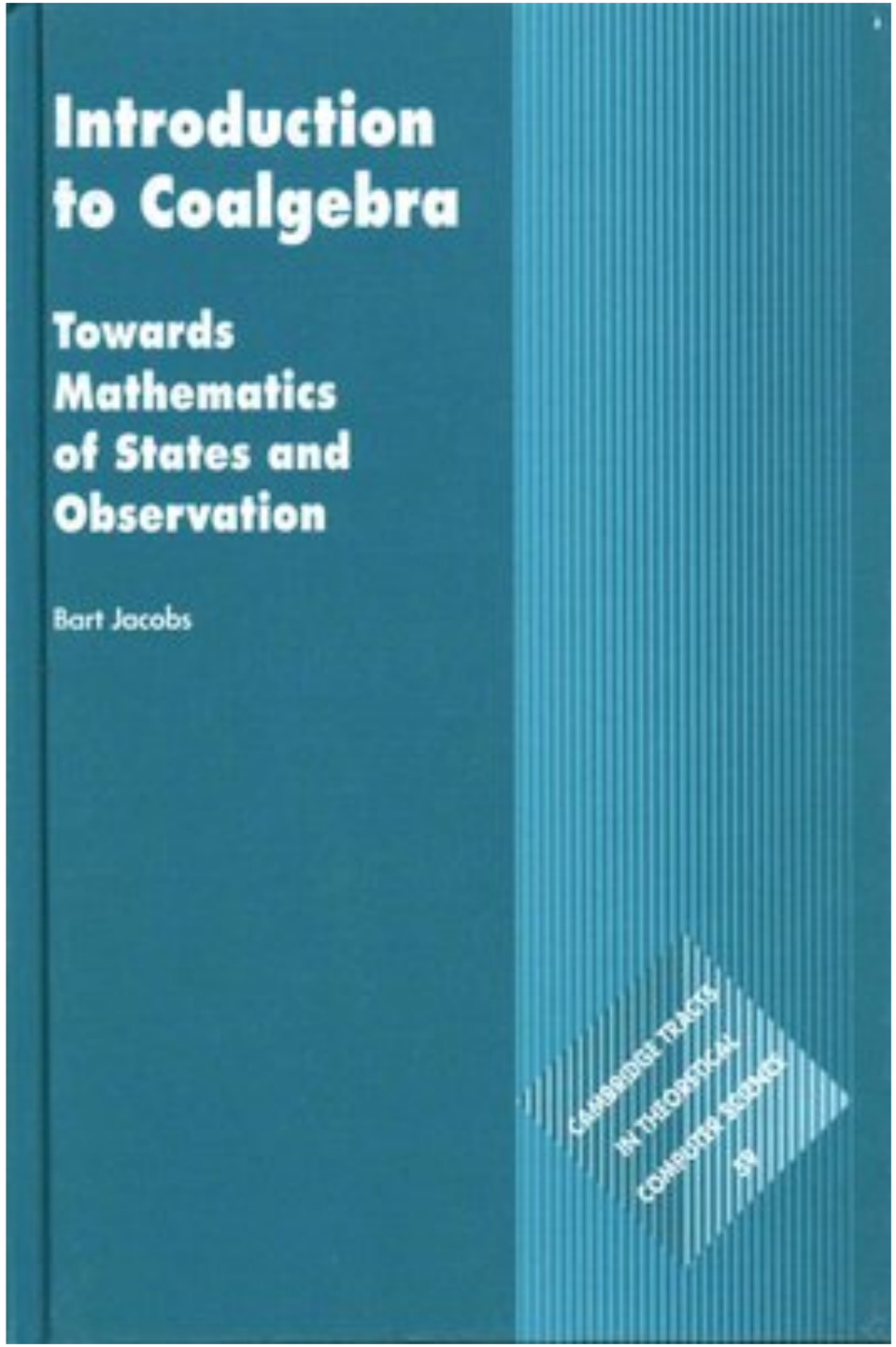
MSC: 68Q10; 68Q55

PACS: D.3; F.1; F.3

Keywords: Coalgebra; Algebra; Dynamical system; Transition system; Bisimulation; Universal coalgebra; Universal algebra; Congruence; Homomorphism; Induction; Coinduction; Variety; Covariety

E-mail address: janr@cwi.nl (J.J.M.M. Rutten).

PII: S0304-3975(00)00056-6



Contents lists available at ScienceDirect

Theoretical Computer Science

journal homepage: www.elsevier.com/locate/tcs

Probabilistic systems coalgebraically: A survey

Ana Sokolova *

Department of Computer Sciences, University of Salzburg, Austria

ARTICLE INFO

Keywords:
Probabilistic systems
Coalgebra
Markov chains
Markov processes

ABSTRACT

We survey the work on both discrete and continuous-space probabilistic systems as coalgebras, starting with how probabilistic systems are modeled as coalgebras and followed by a discussion of their bisimilarity and behavioral equivalence, mentioning results that follow from the coalgebraic treatment of probabilistic systems. It is interesting to note that, for different reasons, for both discrete and continuous probabilistic systems it may be more convenient to work with behavioral equivalence than with bisimilarity.

1. Introduction

Probabilistic systems are models of systems that involve quantitative information about uncertainty. They have been extensively studied in the past two decades in the area of probabilistic verification and concurrency theory. The models originate in the rich theory of Markov chains and Markov processes (see e.g. [49]) and in the early work on probabilistic automata [63,61].

Discrete probabilistic systems, see e.g. [49,77,30,55,62,67,33,22,70] for an overview, are transition systems on discrete state spaces and come in different flavors: fully probabilistic (Markov chains), labeled (with reactive or generative labels), or combining non-determinism and probability. Probabilities in discrete probabilistic systems appear as labels on transitions between states. For example, in a Markov chain a transition from one state to another is taken with a given probability.

Continuous probabilistic systems, see e.g. [7,23,26,11,21,45] as well as the recent books [59,27,28] that contain most of the research on continuous probabilistic systems, are transition systems modeling probabilistic behavior on continuous state spaces. The basic model is that of a Markov process. Central to continuous probabilistic systems is the notion of a probability measure on a measurable space. Therefore, the state space of a continuous probabilistic system is equipped with a σ-algebra and forms a measurable space. It is no longer the case that the probability of moving from one state to another determines the behavior of the system. Actually, the probability of reaching any single state from a given state may be zero while the probability of reaching a subset of states is nonzero. A Markov process is specified by the probability of moving from any source state to any measurable subset in the σ-algebra, which is intuitively interpreted as the probability of moving from the source state to some state in the subset.

Both discrete and continuous probabilistic systems can be modeled as coalgebras and coalgebra theory has proved a useful and fruitful means to deal with probabilistic systems. In this paper, we give an overview of how to model probabilistic systems as coalgebras and survey coalgebraic results on discrete and continuous probabilistic systems. Having modeled probabilistic systems as coalgebras, there are two types of results where coalgebra meets probabilistic systems: (1) particular problems for probabilistic systems have been solved using coalgebraic techniques, and (2) probabilistic systems appear as popular examples on which generic coalgebraic results are instantiated. The results of the second kind are not to be considered of less importance: sometimes they lead to completely new results not known in the community of probabilistic

Coalg_F	F	name for $X \to FX$/reference
MC	$\mathcal{D}$	Markov chains
DLTS	$(\_ + 1)^A$	deterministic automata
LTS	$\mathcal{P}(A \times \_) \cong \mathcal{P}^A$	non-deterministic automata, LTSs
React	$(\mathcal{D} + 1)^A$	reactive systems [55,30]
Gen	$\mathcal{D}(A \times \_) + 1$	generative systems [30]
Str	$\mathcal{D} + (A \times \_) + 1$	stratified systems [30]
Alt	$\mathcal{D} + \mathcal{P}(A \times \_)$	alternating systems [33]
Var	$\mathcal{D}(A \times \_) + \mathcal{P}(A \times \_)$	Vardi systems [77]
SSeg	$\mathcal{P}(A \times \mathcal{D})$	simple Segala systems [67,66]
Seg	$\mathcal{P}\mathcal{D}(A \times \_)$	Segala systems [67,66]
Bun	$\mathcal{D}\mathcal{P}(A \times \_)$	bundle systems [22]
PZ	$\mathcal{P}\mathcal{D}\mathcal{P}(A \times \_)$	Pnueli–Zuck systems [62]
MG	$\mathcal{P}\mathcal{D}\mathcal{P}(A \times \_ + \_)$	most general systems

Fig. 1. Discrete probabilistic system types.

RL algorithms can be explored for these stochastic coalgebras!

Final Coalgebras

- In a category of coalgebras, where each object is X -> F(X), a final coalgebra is an isomorphism X ~ F(X)
- Final coalgebra theorem (Aczel, Mendler): for a wide class of endofunctors, final coalgebras exist (weak pullbacks)
- RL is essentially coinduction in a coalgebra

$$V^\pi = R^\pi + \gamma P^\pi V^\pi = T^\pi(V)$$

MDP Coalgebras

- **Any MDP is defined as a tuple M = (S,A,R,P)**
- **Given any action a, it induces a distribution on next states**
- **Any fixed policy defines an induced Markov chain**
- **Markov chains are coalgebras of the distribution functor D**
  - $\alpha_S^M : S \to^M \mathscr{D}(S)$

Long-Term Values in Markov Decision Processes, (Co)Algebraically

Frank M. V. Feys[1], Helle Hvid Hansen[1], and Lawrence S. Moss[2]

[1] Department of Engineering Systems and Services, TPM, Delft University of Technology, Delft, The Netherlands {f.m.v.feys, h.h.hansen}@tudelft.nl
[2] Department of Mathematics, Indiana University, Bloomington IN, 47405 USA
lsm@cs.indiana.edu

Abstract. This paper studies Markov decision processes (MDPs) from the categorical perspective of coalgebra and algebra. Probabilistic systems, similar to MDPs but without rewards, have been extensively studied, also coalgebraically, from the perspective of program semantics. In this paper, we focus on the role of MDPs as models in optimal planning, where the reward structure is central. The main contributions of this paper are (i) to give a coinductive explanation of policy improvement using a new proof principle, based on Banach's Fixpoint Theorem, that we call contraction coinduction, and (ii) to show that the long-term value function of a policy with respect to discounted sums can be obtained via a generalized notion of corecursive algebra, which is designed to take boundedness into account. We also explore boundedness features of the Kantorovich lifting of the distribution monad to metric spaces.

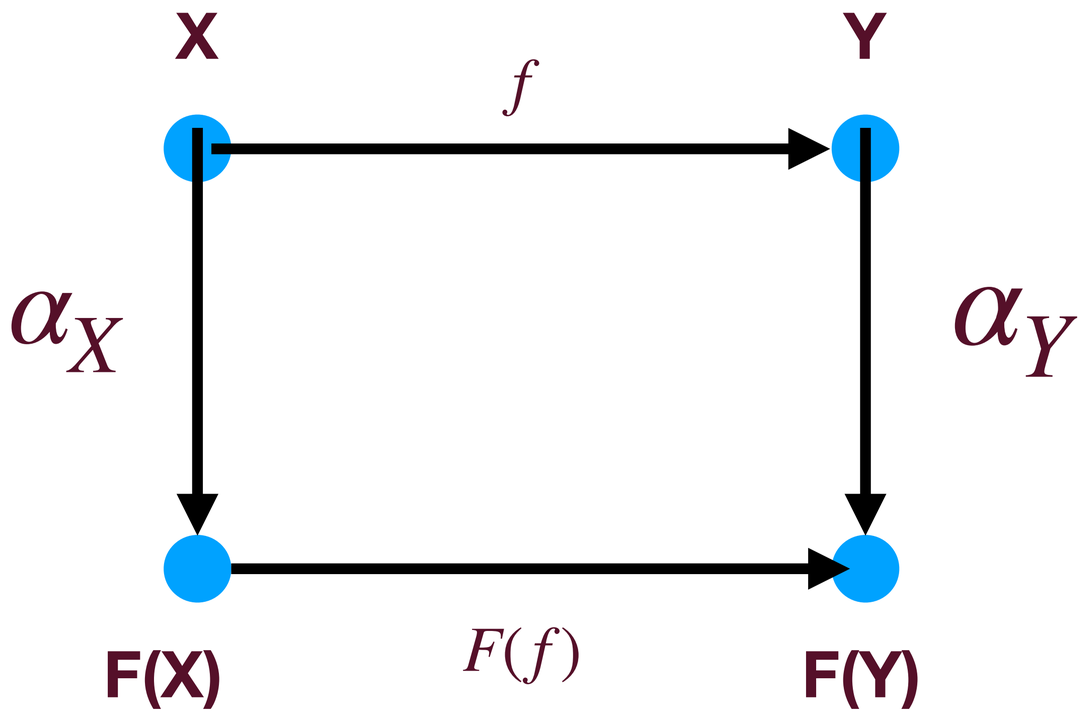
Keywords: Markov decision process · long-term value · discounted sum · coalgebra · algebra · corecursive algebra · fixpoint · metric space.

This paper can be extended to the RL setting

Non-well-founded sets

- **Non-well-founded sets violate the ZFC+ axioms of set theory**
- **In particular, the axiom of well-foundedness states that there cannot be any infinite membership chains**
- **Many sets in computer science are not well-founded**
- **Infinite data structures: lists, trees, recursion, stacks**
- **Many AI problems involve non-well-founded sets**
  - **Common knowledge, causality with feedback, natural language**

Homomorphisms of Coalgebras



MDP homomorphisms are a special case of this framework

RL as Metric Coinduction

APPLICATIONS OF METRIC COINDUCTION

DEXTER KOZEN AND NICHOLAS RUOZZI

Computer Science Department, Cornell University, Ithaca, NY 14853-7501, USA
e-mail address: kozen@cs.cornell.edu

Computer Science Department, Yale University, New Haven, CT 06520-8285, USA
e-mail address: Nicholas.Ruozzi@yale.edu

Abstract. Metric coinduction is a form of coinduction that can be used to establish properties of objects constructed as a limit of finite approximations. One can prove a coinduction step showing that some property is preserved by one step of the approximation process, then automatically infer by the coinduction principle that the property holds of the limit object. This can often be used to avoid complicated analytic arguments involving limits and convergence, replacing them with simpler algebraic arguments. This paper examines the application of this principle in a variety of areas, including infinite streams, Markov chains, Markov decision processes, and non-well-founded sets. These results point to the usefulness of coinduction as a general proof technique.

1. Introduction

Mathematical induction is firmly entrenched as a fundamental and ubiquitous proof principle for proving properties of inductively defined objects. Mathematics and computer science abound with such objects, and mathematical induction is certainly one of the most important tools, if not the most important, at our disposal.

Perhaps less well entrenched is the notion of coinduction. Despite recent interest, coinduction is still not fully established in our collective mathematical consciousness. A contributing factor is that coinduction is often presented in a relatively restricted form. Coinduction is often considered synonymous with bisimulation and is used to establish equality or other relations on infinite data objects such as streams [20] or recursive types [11].

$$\frac{\exists u\ \varphi(u) \qquad \forall u\ \varphi(u) \Rightarrow \varphi(H(u))}{\varphi(u^*)}$$

Contraction mapping convergence in MDPs

is a special case of metric coinduction

Induction vs Coinduction

- Given the class of all (non)well-founded sets
  - X —> F(X) is the powerset coalgebra
  - F(X) —> X is the powerset algebra
- The initial object in the category of algebras is well-founded sets
- The final object in the category of coalgebras is non-well-founded sets

Final Coalgebras

- A final object in a category is defined as one for which there is a unique morphism into it from any other object
- In the category of coalgebras, the final object is called a final coalgebra
- Example: in the coalgebra of finite state automata, the final coalgebra is the smallest automaton accepting a language
- Example: in the coalgebra of MDPs, the final coalgebra is the smallest MDP that defines the optimal value function

Lambek's Lemma

Definition 83. An F-coalgebra (A, α) is a *fixed point* for F, written as $A \simeq F(A)$ if α is an isomorphism between A and $F(A)$. That is, not only does there exist an arrow $A \to F(A)$ by virtue of the coalgebra α, but there also exists its inverse $\alpha^{-1} : F(A) \to A$ such that

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

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

Theorem 23. Lambek: A final F-coalgebra is a fixed point of the endofunctor F.

Math. Struct. in Comp. Science (2005), *vol.* 15, *pp.* 409–432.
doi:10.1017/S0960129505004731 Printed in the United Kingdom

A general final coalgebra theorem

JIŘÍ ADÁMEK[†], STEFAN MILIUS[‡] and JIŘÍ VELEBIL[§]

[†‡]*Institute of Theoretical Computer Science, Technical University of Braunschweig, Germany*
E-mail: {adamek,milius}@iti.cs.tu-bs.de
[§]*Faculty of Electrical Engineering, Czech Technical University, Prague*

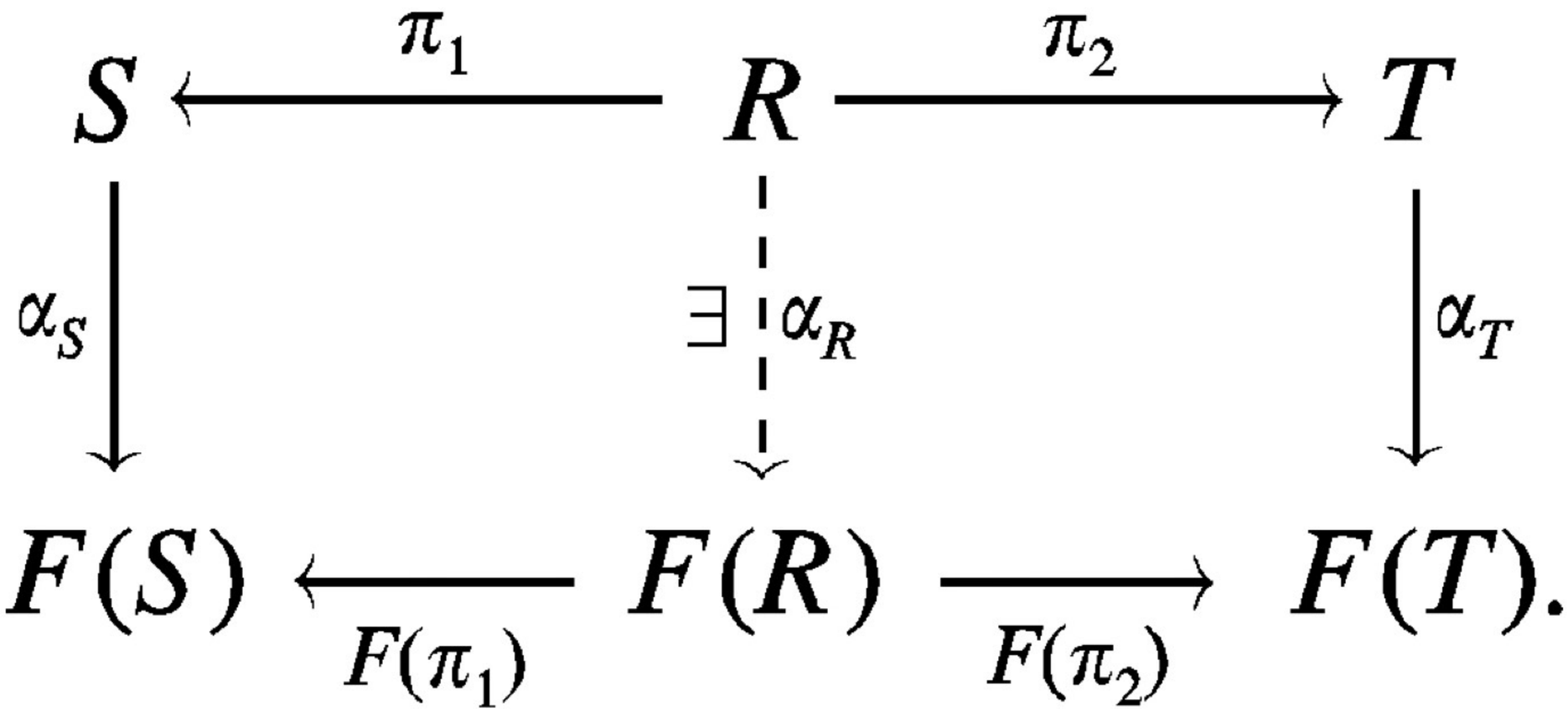
Received 27 March 2003

By the Final Coalgebra Theorem of Aczel and Mendler, every endofunctor of the category of sets has a final coalgebra, which, however, may be a proper class. We generalise this to all 'well-behaved' categories $\mathscr{K}$. The role of the category of classes is played by a free cocompletion $\mathscr{K}^\infty$ of $\mathscr{K}$ under transfinite colimits, that is, colimits of ordinal-indexed chains. Every endofunctor F of $\mathscr{K}$ has a canonical extension to an endofunctor F^∞ of $\mathscr{K}^\infty$ which is proved to have a final coalgebra (and an initial algebra). Based on this, we prove a general solution theorem: for every endofunctor of a locally presentable category $\mathscr{K}$ all guarded equation-morphisms have unique solutions. The last result does not need the extension $\mathscr{K}^\infty$: the solutions are always found within the category $\mathscr{K}$.

Occam's Razor Coalgebraically

- We can now define a coalgebraic version of Occam's Razor
- Given any category of coalgebras, where there is a final coalgebra
- Any other coalgebra must define a unique morphism into the final coalgebra
- If this unique morphism is injective (or a monomorphism), the given coalgebra must be minimal
- States of the final coalgebra define ``behaviors" (see Jacobs book)

Bisimulation in Coalgebras



An Extrinsic Theory of AGI

"I propose to consider the question, 'Can machines think'?" – *Alan Turing, Mind, Volume LIX, Issue 236, October 1950, Pages 433–460.*

UNIVERSAL IMITATION GAMES*

A PREPRINT

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

May 6, 2024

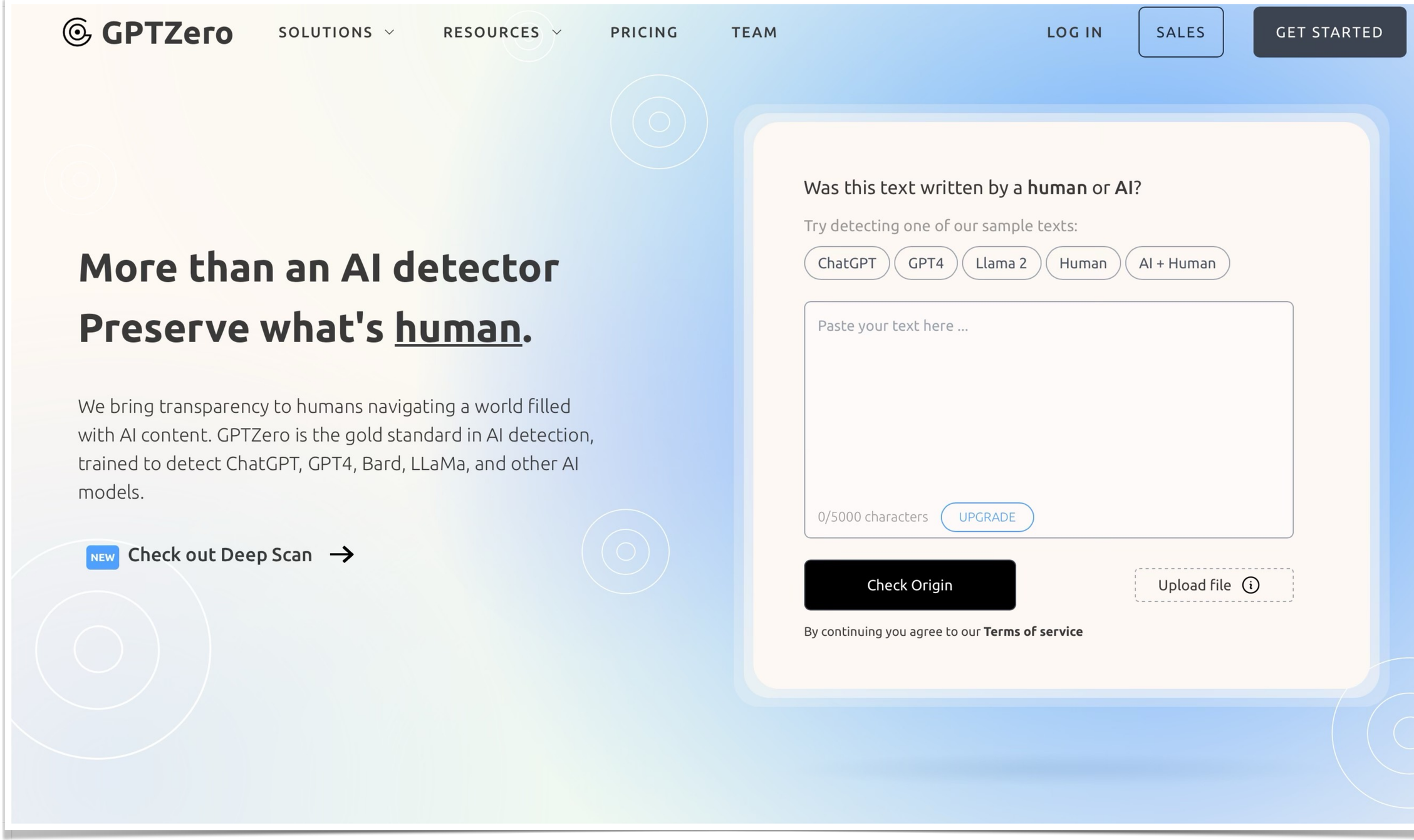
ABSTRACT

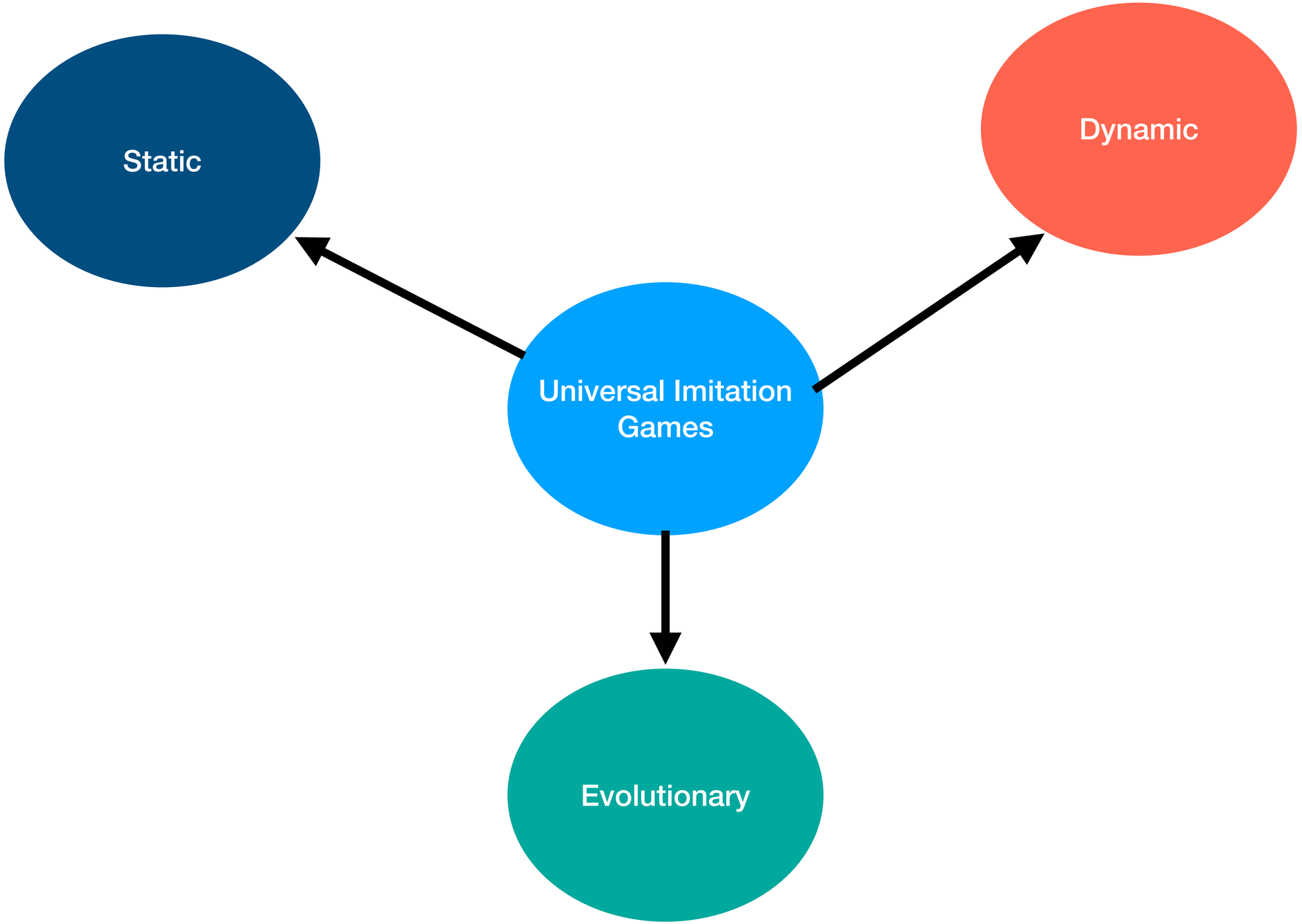
In 1950, Alan Turing proposed a framework called an *imitation game* in which the participants are to be classified `Human or Machine` solely from natural language interactions. Using mathematics largely developed since Turing – category theory – we investigate a broader class of *universal imitation games* (UIGs). Choosing a category means defining a collection of objects and a collection of composable arrows between each pair of objects that represent "measurement probes" for solving UIGs. The theoretical foundation of our paper rests on two celebrated results by Yoneda. The first, called the Yoneda Lemma, discovered in 1954 – the year of Turing's death – shows that objects in categories can be identified up to isomorphism solely with measurement probes defined by composable arrows. Yoneda embeddings are universal representers of objects in categories. A simple yet general solution to the static UIG problem, where the participants are not changing during the interactions, is to determine if the Yoneda embeddings are (weakly) isomorphic. A *universal property* in category theory is defined by an *initial* or *final* object. A second foundational result of Yoneda from 1960 defines initial objects called *coends* and final objects called *ends*, which yields a categorical "integral calculus" that unifies probabilistic generative models, distance-based kernel, metric and optimal transport models, as well as topological manifold representations. When participants adapt during interactions, we study two special cases: in *dynamic UIGs*, "learners" imitate "teachers". We contrast the initial object framework of *passive learning from observation* over well-founded sets using inductive inference – extensively studied by Gold, Solomonoff, Valiant, and Vapnik – with the final object framework of *coinductive inference* over non-well-founded sets and universal coalgebras, which formalizes learning from *active experimentation* using causal inference or reinforcement learning. We define a category-theoretic notion of minimum description length or Occam's razor based on final objects in coalgebras. Finally, we explore *evolutionary UIGs*, where a society of participants is playing a large-scale imitation game. Participants in evolutionary UIGs can go extinct from "birth-death" evolutionary processes that model how novel technologies or "mutants" disrupt previously stable equilibria. Given the rapidly rising energy costs of playing imitation games on classical computers, it seems likely that tomorrow's imitation games may have to played on non-classical computers. We end with a brief discussion of how our categorical framework extends to imitation games on quantum computers.

Keywords AI · Imitation Games · Category Theory · Evolution · Game Theory · Machine Learning · Quantum Computing

"I propose to consider the question, 'Can machines think'?" – *Alan Turing, Mind, Volume LIX, Issue 236, October 1950, Pages 433–460.*

Universal Imitation Games


GPTZero
SOLUTIONS
RESOURCES
PRICING
TEAM
LOG IN
SALES
GET STARTED
More than an AI detector
Preserve what's human.
We bring transparency to humans navigating a world filled with AI content. GPTZero is the gold standard in AI detection, trained to detect ChatGPT, GPT4, Bard, LLaMa, and other AI models.
NEW
Check out Deep Scan
Was this text written by a human or AI?
Try detecting one of our sample texts:
ChatGPT
GPT4
Llama 2
Human
AI + Human
Paste your text here ...
0/5000 characters
UPGRADE
Check Origin
Upload file
By continuing you agree to our Terms of service


Static
Dynamic
Universal Imitation Games
Evolutionary

UNIVERSAL IMITATION GAMES*

A PREPRINT

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

May 6, 2024

ABSTRACT

In 1950, Alan Turing proposed a framework called an *imitation game* in which the participants are to be classified `Human or Machine` solely from natural language interactions. Using mathematics largely developed since Turing – category theory – we investigate a broader class of *universal imitation games* (UIGs). Choosing a category means defining a collection of objects and a collection of composable arrows between each pair of objects that represent "measurement probes" for solving UIGs. The theoretical foundation of our paper rests on two celebrated results by Yoneda. The first, called the Yoneda Lemma, discovered in 1954 – the year of Turing's death – shows that objects in categories can be identified up to isomorphism solely with measurement probes defined by composable arrows. Yoneda embeddings are universal representers of objects in categories. A simple yet general solution to the static UIG problem, where the participants are not changing during the interactions, is to determine if the Yoneda embeddings are (weakly) isomorphic. A *universal property* in category theory is defined by an *initial* or *final* object. A second foundational result of Yoneda from 1960 defines initial objects called *coends* and final objects called *ends*, which yields a categorical "integral calculus" that unifies probabilistic generative models, distance-based kernel, metric and optimal transport models, as well as topological manifold representations. When participants adapt during interactions, we study two special cases: in *dynamic UIGs*, "learners" imitate "teachers". We contrast the initial object framework of *passive learning from observation* over well-founded sets using inductive inference – extensively studied by Gold, Solomonoff, Valiant, and Vapnik – with the final object framework of *coinductive inference* over non-well-founded sets and universal coalgebras, which formalizes learning from *active experimentation* using causal inference or reinforcement learning. We define a category-theoretic notion of minimum description length or Occam's razor based on final objects in coalgebras. Finally, we explore *evolutionary UIGs*, where a society of participants is playing a large-scale imitation game. Participants in evolutionary UIGs can go extinct from "birth-death" evolutionary processes that model how novel technologies or "mutants" disrupt previously stable equilibria. Given the rapidly rising energy costs of playing imitation games on classical computers, it seems likely that tomorrow's imitation games may have to played on non-classical computers. We end with a brief discussion of how our categorical framework extends to imitation games on quantum computers.

Extrinsic AGI Framework

An Intrinsic Theory of AGI

What is our AGI Framework?

- Each AGI modality defines a category called a topos
  - Objects in the category define entities, and arrows specify their interaction
- An AGI “mind” has to synthesize multiple AGI modalities
  - Learning to build a causal model
  - Planning to take optimal actions
  - Communication with language
- Topos theory shows us how to build AGI minds

Set Theory vs Topos Theory

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

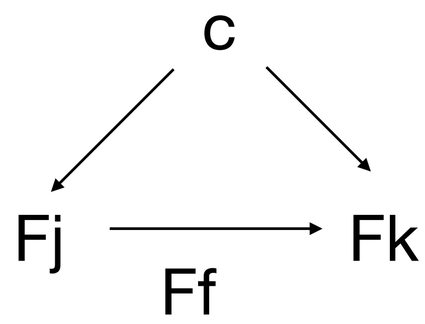
Topos as a Category

- A topos is a category that
  - Has all limits and colimits
  - Has exponential objects or an internal hom-object
  - Subobject classifier

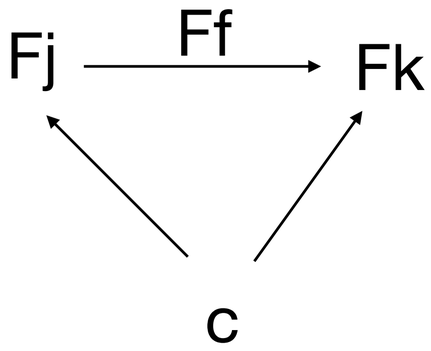
Limits and Colimits

- Limits and colimits are abstractions of common mathematical constructs
  - Limits: meet, max, supremum, product,…
  - Colimits: join, min, infimum, coproduct,…
- They are defined by universal properties
  - A limit is a final object in a category of diagrams
  - A colimit is an initial object in a category of diagrams

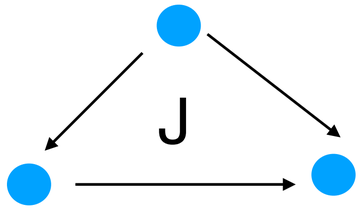
Limits and colimits: Diagrams F: J —> C

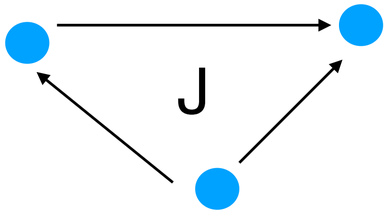


Cone under c



Cone over c





A limit is an object LimF in C

Universal cone under LimF

A colimit is an object colimF in C

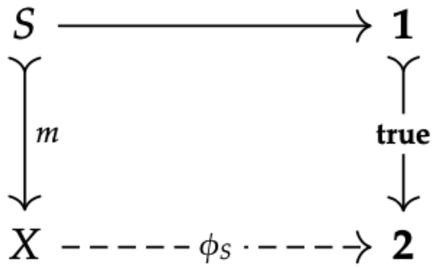
Universal cone over colimF

Exponential Objects

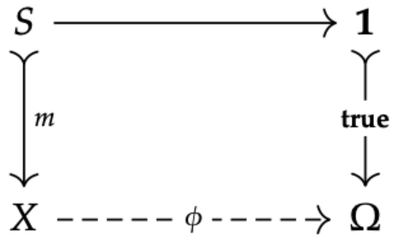
- Given a category C with products:
  - For each object c: define a functor F: c * ? —> c: C —> C
  - If functor F has a right adjoint G: $(-)^c$: C —> C
  - We say C has exponential objects

Subobject Classifier

Set theory



Topos theory



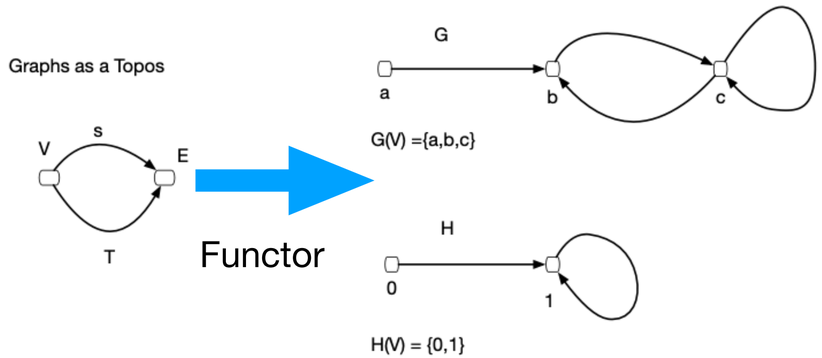
Let X = natural numbers

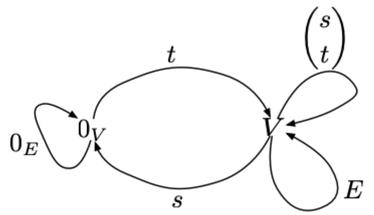
Let S = prime numbers

X is an arbitrary object, S is a subobject:

Causal model, deep learning model

Topos of Graphs





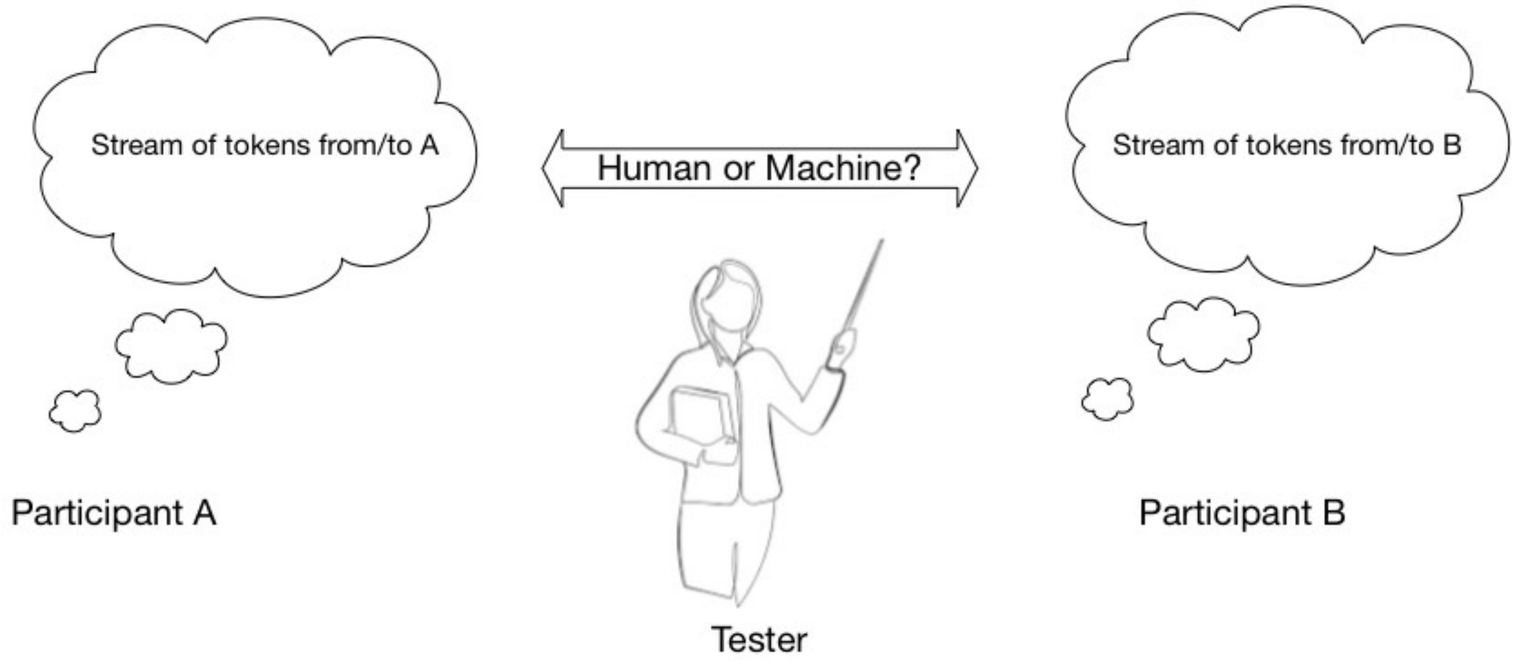
Subobject Classifer

Yoneda embedding:

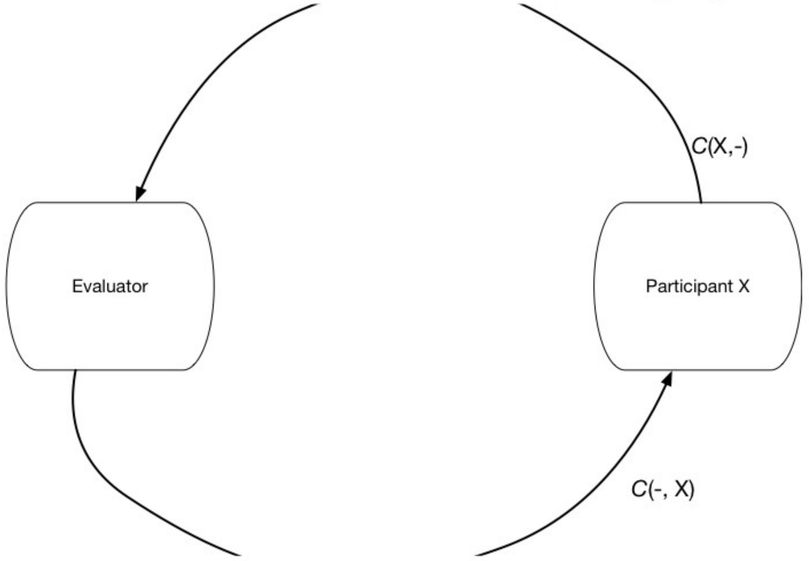
C(-, V): $\{1_V\}$

C(-, E): $\{s, t, 1_E\}$

[Vigna, Arxiv, 2003]



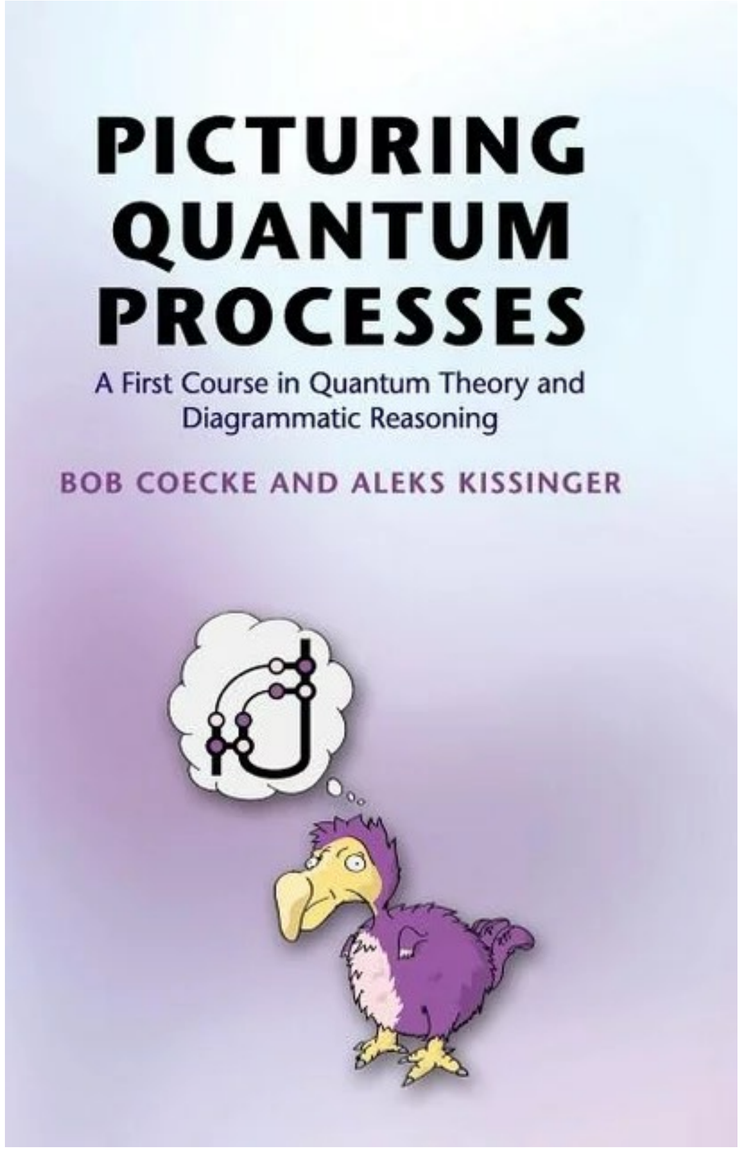
Count me out on this one. I never could write poetry.



Please write me a sonnet on the subject of the Forth Bridge.

Quantum Computing and AGI

Quantum Computing in Categories



A categorical semantics of quantum protocols

Samson Abramsky and Bob Coecke

Oxford University Computing Laboratory,
Wolfson Building, Parks Road, Oxford OX1 3QD, UK.
samson.abramsky · bob.coecke@comlab.ox.ac.uk

Abstract

We study quantum information and computation from a novel point of view. Our approach is based on recasting the standard axiomatic presentation of quantum mechanics, due to von Neumann [28]*, at a more abstract level, of compact closed categories with biproducts. We show how the essential structures found in key quantum information protocols such as teleportation* [5]*, logic-gate teleportation* [12]*, and entanglement swapping* [29] *can be captured at this abstract level. Moreover, from the combination of the — apparently purely qualitative — structures of compact closure and biproducts there emerge 'scalars' and a 'Born rule'. This abstract and structural point of view opens up new possibilities for describing and reasoning about quantum systems. It also shows the degrees of axiomatic freedom: we can show what requirements are placed on the (semi)ring of scalars* $\mathbf{C}(\mathrm{I},\mathrm{I})$*, where* $\mathbf{C}$ *is the category and* I *is the tensor unit, in order to perform various protocols such as teleportation. Our formalism captures both the information-flow aspect of the protocols* [8, 9]*, and the*

tation [12], and *entanglement swapping* [29]. The ideas illustrated in these protocols form the basis for novel and potentially very important applications to secure and fault-tolerant communication and computation [7, 12, 20].

We now give a thumbnail sketch of teleportation to motivate our introductory discussion. (A more formal 'standard' presentation is given in Section 2. The — radically different — presentation in our new approach appears in Section 9.) Teleportation involves using an entangled pair of qubits (q_A, q_B) as a kind of communication channel to transmit an unknown qubit q from a source A ('Alice') to a remote target B ('Bob'). A has q and q_A, while B has q_B. We firstly entangle q_A and q at A (by performing a suitable unitary operation on them), and then perform a measurement on q_A and q.[1] This forces a 'collapse' in q_B because of its entanglement with q_A. We then send two classical bits of information from A to B, which encode the four possible results of the measurement we performed on q and q_A. Based on this classical communication, B then performs a 'correction' by applying one of four possible operations (unitary transformations) to q_B, after which q_B *has*

DisCoPy

The Python toolkit for computing with string diagrams

DisCoPy is a Python toolkit for computing with string diagrams.

- Documentation: https://docs.discopy.org
- Repository: https://github.com/discopy/discopy

Why?

Applied category theory is information plumbing. It's boring… but *plumbers save lives than doctors*.

As string diagrams become as ubiquitous as matrices, they need their own funda package: *DisCoPy*.

How?

DisCoPy began as an implementation of:

- DisCoCat (distributional compositional categorical) models,
- and QNLP (quantum natural language processing).

This application has now been packaged into its own library, lambeq.

Who?

- Giovanni de Felice (CEO)
- Alexis Toumi (COO)
- Richie Yeung (CFO)
- Boldizsár Poór (CTO)
- Bob Coecke (Honorary President)

Diagrammatic Differentiation for Quantum Machine Learning

Alexis Toumi[⋆†], Richie Yeung[†], Giovanni de Felice[⋆†]

⋆ Department of Computer Science, University of Oxford † Cambridge Quantum Computing Ltd.

We introduce diagrammatic differentiation for tensor calculus by generalising the dual number construction from rigs to monoidal categories. Applying this to ZX diagrams, we show how to calculate diagrammatically the gradient of a linear map with respect to a phase parameter. For diagrams of parametrised quantum circuits, we get the well-known parameter-shift rule at the basis of many variational quantum algorithms. We then extend our method to the automatic differentation of hybrid classical-quantum circuits, using diagrams with bubbles to encode arbitrary non-linear operators. Moreover, diagrammatic differentiation comes with an open-source implementation in DisCoPy, the Python library for monoidal categories. Diagrammatic gradients of classical-quantum circuits can then be simplified using the PyZX library and executed on quantum hardware via the tket compiler. This opens the door to many practical applications harnessing both the structure of string diagrams and the computational power of quantum machine learning.

Introduction

String diagrams are a graphical language introduced by Penrose [33] to manipulate tensor expressions: wires represent vector spaces, nodes represent multi-linear maps between them. In [34], these diagrams are used to describe the geometry of space-time and an extra piece of notation is introduced: the covariant derivative is represented as a bubble around the tensor to be differentiated. Joyal and Street [25, 26] characterised string diagrams as the arrows of free monoidal categories, however their geometry of tensor

Meaning arises out of words in a sentence using ‘quantum entanglement

Category Theory for Quantum Natural Language Processing

Alexis TOUMI

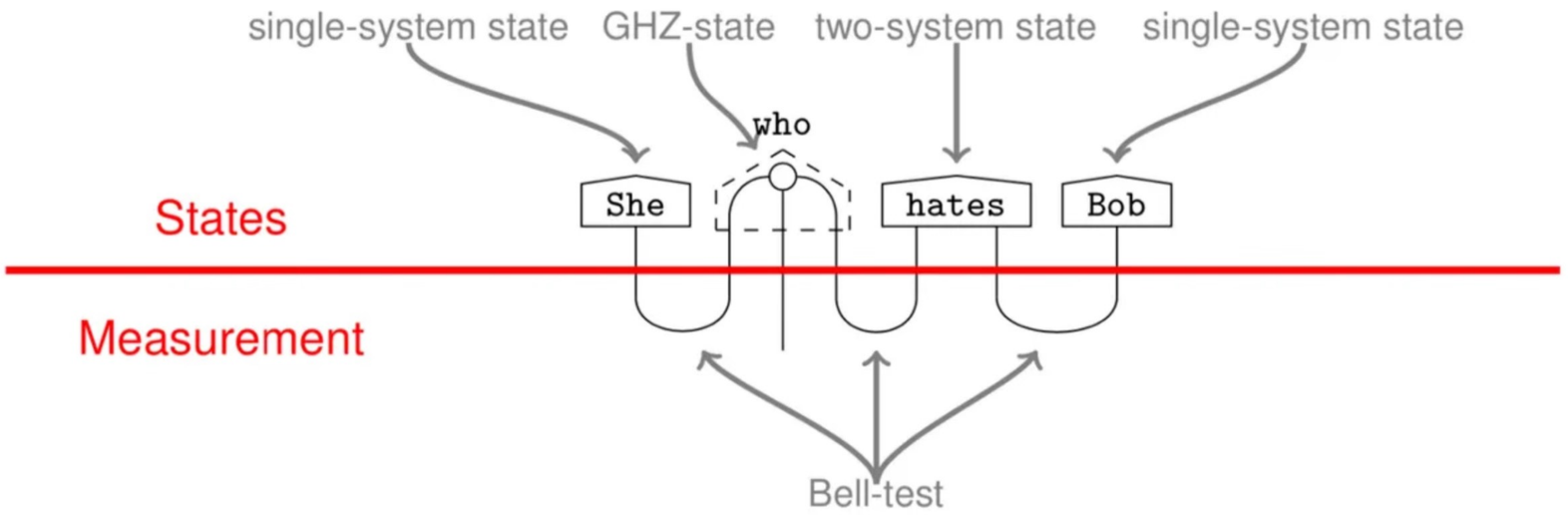
Wolfson College

University of Oxford

A thesis submitted for the degree of

Doctor of Philosophy

Trinity 2022



https://medium.com/cambridge-quantum-computing/quantum-natural-language-processing-748d6f27b31d

DisCoPy

The Python toolkit for computing with string diagrams

DisCoPy is a Python toolkit for computing with string diagrams.

- Documentation: https://docs.discopy.org
- Repository: https://github.com/discopy/discopy

Why?

Applied category theory is information plumbing. It's boring… but *plumbers save more lives than doctors*.

As string diagrams become as ubiquitous as matrices, they need their own fundamental package: *DisCoPy*.

How?

DisCoPy began as an implementation of:

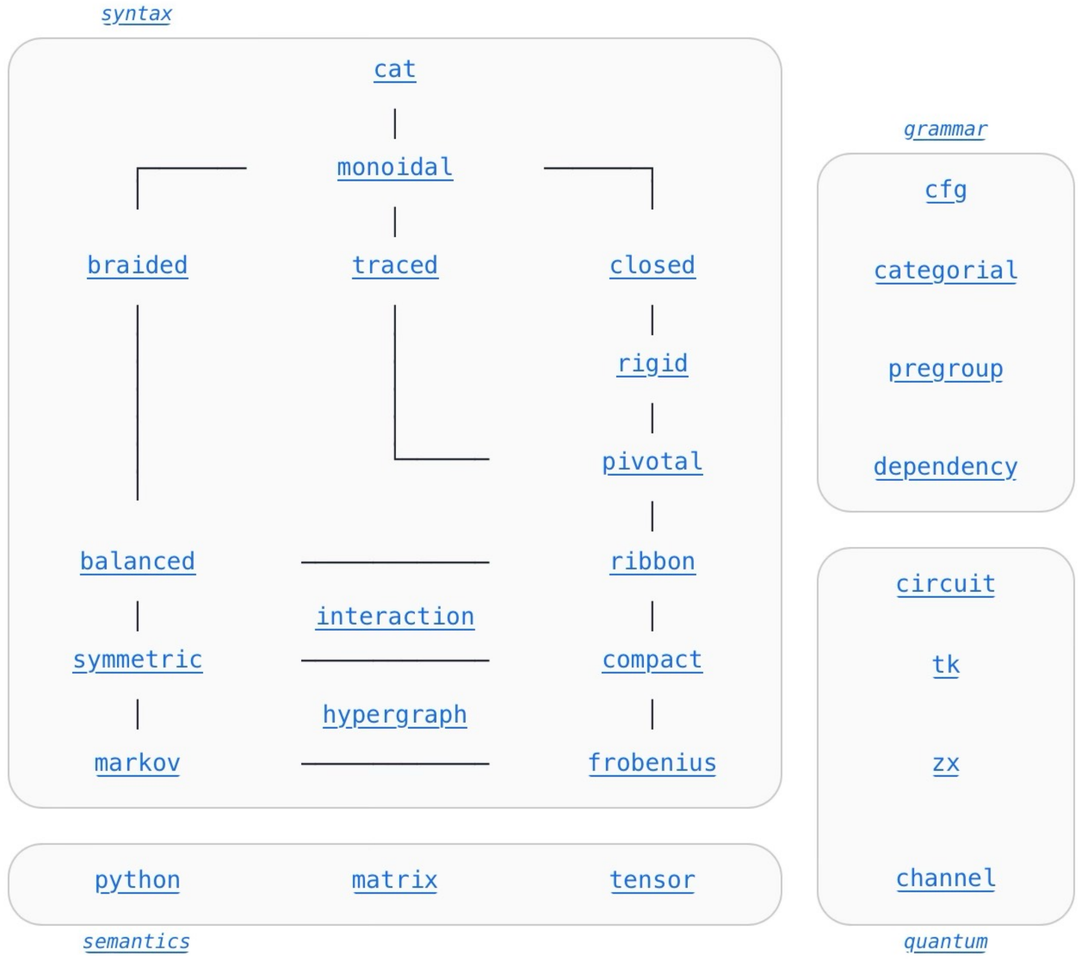
- DisCoCat (distributional compositional categorical) models,
- and QNLP (quantum natural language processing).

This application has now been packaged into its own library, lambeq.

Who?

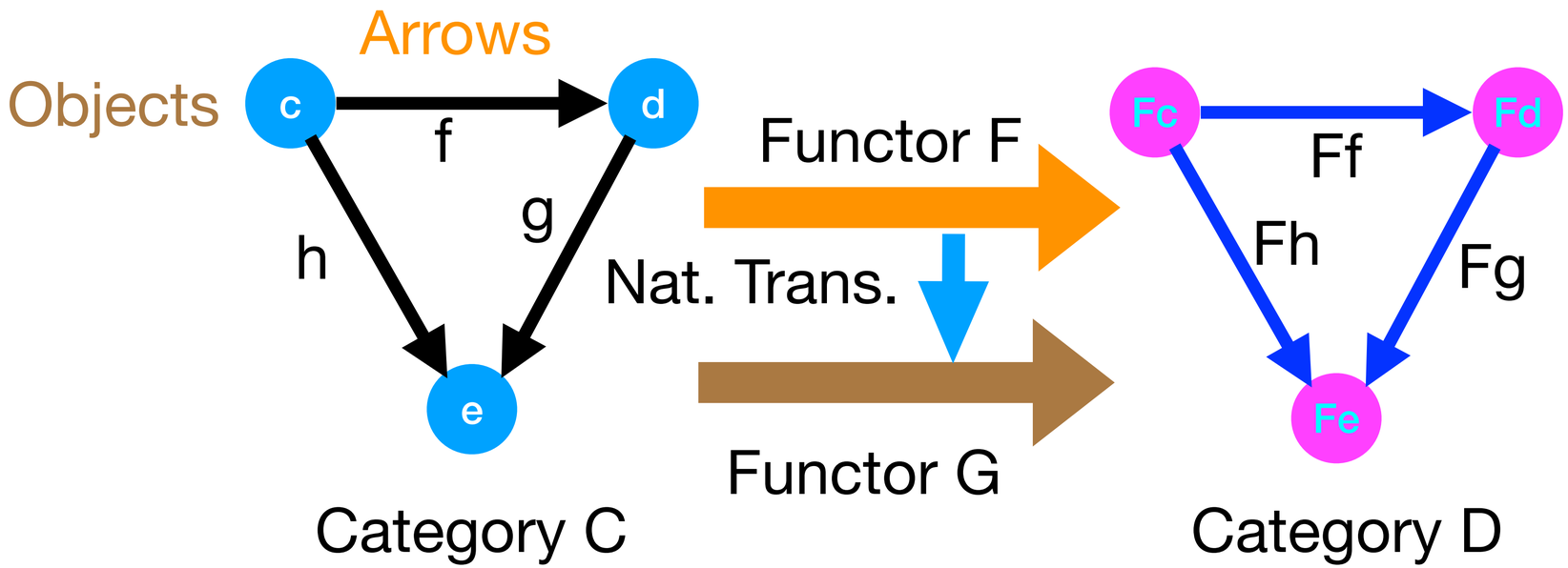
- Giovanni de Felice (CEO)
- Alexis Toumi (COO)
- Richie Yeung (CFO)
- Boldizsár Poór (CTO)
- Bob Coecke (Honorary President)

Want to contribute or just ask us a question? Get in touch on Discord!



Summary

The Building Blocks



Category Theory is all about compositionality

The category of all categories is a category!

How to think with functors

- Abstraction and Compositionality
  - What are the generic objects and arrows of a category?
  - What universal properties does the category satisfy?
- How to define adjoint functors between a category and other categories?
- How to convert a category into a simplicial set or a topos?
  - Yoneda Embeddings: x —> C(-, x)

Many Applications in A(G)I

- Machine Learning
  - New ways to model deep learning, manifold learning clustering
- Probabilistic and causal inference
  - Markov categories show how to reason with string diagrams
- Reinforcement learning over universal coalgebras
  - A new way to think about state using functors
- A new compositional framework for AGI