Categorical Foundation of Explainable AI

J. Eickhoff

Lecture: Explainable Artificial Intelligence, 10/20/2025

Inhalt

Category Theory

"Mathematics is the art of giving the same name to different things."

– Henri Poincaré, Science et Méthode (1908)

Category Theory

Category Theory provides a unifying mathematical language for describing structures and relationships between them.
It abstracts the essential idea of *composition* and *identity* found across all areas of mathematics.

Idea

"Mathematics is about structures, and category theory is the study of structure itself."

A *category* consists of **objects** and **morphisms** (arrows) between them, with composition and identity satisfying associativity and unity laws.

Category Theory

Definition (Category)

A *category* $\mathcal{C}$ consists of the following components:

- a class of **objects**, denoted by $\mathrm{Ob}(\mathcal{C})$,
- for each pair of objects $X, Y \in \mathrm{Ob}(\mathcal{C})$, a class of **morphisms**

$$\mathrm{Hom}_{\mathcal{C}}(X, Y),$$

  which includes a *identity morphism* $\mathrm{Id}_X \in \mathrm{Hom}_{\mathcal{C}}(X, X)$ for each object X,
- and a **composition** operation

$$\circ \colon \mathrm{Hom}_{\mathcal{C}}(X, Y) \times \mathrm{Hom}_{\mathcal{C}}(Y, Z) \to \mathrm{Hom}_{\mathcal{C}}(X, Z),$$

  satisfying

$$\mathrm{Id}_Y \circ f = f = f \circ \mathrm{Id}_X, \ (h \circ g) \circ f = h \circ (g \circ f).$$

Category Theory

Definition (Opposite Category)

Let $\mathcal{C}$ be a category. Then $\mathcal{C}^{\mathsf{op}}$ is the *opposite category* of $\mathcal{C}$ with following data:

- $\mathsf{Ob}(\mathcal{C}^{\mathsf{op}}) = \mathsf{Ob}(\mathcal{C})$
- $\mathrm{Hom}_{\mathcal{C}^{\mathrm{op}}}(X, Y) = \mathrm{Hom}_{\mathcal{C}}(Y, X)$

Remark: We call a category *small* if the objects and morphisms form sets. In case the class of morphisms form sets, we call the category *locally small*.

Category Theory

Definition (Functor)

A functor $F\colon \mathcal{C} \to \mathcal{D}$ assigns:

- to each object $X \in \mathcal{C}$ an object $F(X) \in \mathcal{D}$,
- to each morphism $f\colon X \to Y$ in $\mathcal{C}$ a morphism

$$F(f)\colon F(X) \to F(Y)$$

  in $\mathcal{D}$,

such that:

- $F(\mathrm{Id}_X) = \mathrm{Id}_{F(X)}$ for every object X,
- $F(g \circ f) = F(g) \circ F(f)$ for all composable morphisms f, g.

Category Theory

Definition (Natural Transformation)

Let $F, G : \mathcal{C} \to \mathcal{D}$ be functors. A *natural transformation* $\eta : F \Rightarrow G$ assigns to each object $X \in \mathcal{C}$ a morphism

$$\eta_X : F(X) \to G(X)$$

in $\mathcal{D}$ such that for every morphism $f : X \to Y$ in $\mathcal{C}$ the following diagram commutes:

$$\begin{tikzcd} F(X) \arrow[r, "F(f)"] \arrow[d, "\eta_X"'] & F(Y) \arrow[d, "\eta_Y"] \\ G(X) \arrow[r, "G(f)"] & G(Y) \end{tikzcd}$$

If each η_X is an isomorphism, then η is called a *natural isomorphism*.

Category Theory

Definition (Equivalence of Categories)

Two categories $\mathcal{C}$ and $\mathcal{D}$ are said to be *equivalent* if there exist functors

$$F : \mathcal{C} \to \mathcal{D} \quad \text{and} \quad G : \mathcal{D} \to \mathcal{C}$$

such that the compositions

$$G \circ F \cong \mathrm{Id}_{\mathcal{C}} \quad \text{and} \quad F \circ G \cong \mathrm{Id}_{\mathcal{D}}$$

are naturally isomorphic to the respective identity functors.

Inhalt

Institutional model theory

Institutional Model Theory is an abstract framework for logic and model theory.
It provides a *unified categorical description* of various logical systems (e.g. first-order logic, modal logic, algebraic specification).

Idea

"Truth is invariant under change of notation."

That is, the relation between syntax (sentences) and semantics (models) remains consistent across different signatures.

Signature

Definition (Signature)

A *signature* is a quadruple

$$\sum = (\mathcal{S}_{\text{func}}, \mathcal{S}_{\text{rel}}, \mathcal{S}_{\text{const}}, \mathrm{ar}),$$

where

- $\mathcal{S}_{\text{func}}$ and $\mathcal{S}_{\text{rel}}$ are disjoint sets of
  - function symbols (e.g., $+, \times$),
  - relation symbols or predicates (e.g., $\leq, \in$),
  - constant symbols (e.g., $0, 1$),
- and $\mathrm{ar} : \mathcal{S}_{\text{func}} \cup \mathcal{S}_{\text{rel}} \to \mathbb{N}$ assigns a natural number (called the *arity*) to each function or relation symbol.

A function or relation symbol is called *n-ary* if its arity is *n*.

Institutional Model Theory

Let $\mathsf{Cat}^{\mathrm{op}}$ denote the opposite category of small categories. Consider

- a category Sign of *signatures*;
- a functor $\mathrm{Sen} : \mathsf{Sign} \to \mathsf{Set}$ assigning to each Σ its sentences $\mathrm{Sen}(\Sigma)$;
- a functor $\mathrm{Mod} : \mathsf{Sign} \to \mathsf{Cat}^{\mathrm{op}}$ assigning to each Σ its category of models $\mathrm{Mod}(\Sigma)$;
- a satisfaction relation

$$\models_\Sigma \subseteq \ |\mathrm{Mod}(\Sigma)| \times \mathrm{Sen}(\Sigma).$$

Institution Model Theory

Definition (Institution)

An **institution** is a quadruple

$$(\mathsf{Sign}, \mathrm{Sen}, \mathrm{Mod}, \models).$$

For a signature morphism $\sigma : \Sigma \to \Sigma'$:

- **sentence translation** $\mathrm{Sen}(\sigma) : \mathrm{Sen}(\Sigma) \to \mathrm{Sen}(\Sigma')$,
- **reduct functor** $\mathrm{Mod}(\sigma) : \mathrm{Mod}(\Sigma') \to \mathrm{Mod}(\Sigma)$ (contravariant via $\mathsf{Cat}^{\mathrm{op}}$).

Satisfaction condition: for all $M' \in \mathrm{Mod}(\Sigma')$ and $\varphi \in \mathrm{Sen}(\Sigma)$,

$$M' \models_{\Sigma'} \mathrm{Sen}(\sigma)(\varphi) \iff \mathrm{Mod}(\sigma)(M') \models_{\Sigma} \varphi.$$

Example: Institution of Propositional Logic

Propositional Logic can be represented as an institution where:

- **Signatures:** $\Sigma = \{a, b, c, \neg, \wedge, \vee, \rightarrow, \top, \bot\}$ with arity:

$$\mathrm{ar}(\neg) = 1, \quad \mathrm{ar}(\wedge) = \mathrm{ar}(\vee) = \mathrm{ar}(\rightarrow) = 2, \quad \mathrm{ar}(a) = \mathrm{ar}(b) = \mathrm{ar}(c) = 0$$

- **Sentences:** $\mathrm{Sen}(\Sigma)$ – all Boolean formulas over Σ
- **Models:** $\mathrm{Mod}(\Sigma) = \{0, 1\}^{\{a,b,c\}}$
- $M \models_\Sigma \varphi$ iff φ evaluates to true under M

Idea

Truth depends only on Boolean combinations of fixed atomic propositions.

Explicit Example

For $M(a) = 1$, $M(b) = 1$, $M(c) = 0$ we have:

$$(a \wedge b) \rightarrow \neg c = (1 \wedge 1) \rightarrow \neg 0 = 1 \rightarrow 1 = 1.$$

Hence $M \models_\Sigma \varphi$.

Example: Institution of First-Order Logic

First-Order Logic extends propositional logic with variables, quantifiers, and structure.

- Sign – first-order signatures with functions and relations (e.g. $+, 0, <$),
- $\mathrm{Sen}(\Sigma)$ – well-formed formulas over Σ (e.g. $\forall x\,(x + 0 = x)$),
- $\mathrm{Mod}(\Sigma)$ – Σ-structures interpreting symbols, e.g. $(\mathbb{N}, +, 0, <)$,
- $M \models_\Sigma \varphi$ iff φ is true in M under all variable assignments.

Idea

Truth now depends on the interpretation of symbols in a structure

Explicit Example

In $M = (\mathbb{N}, +, 0, <)$:

$$\forall x\,(x + 0 = x)$$

means "for all natural numbers x, $x + 0 = x$". This holds for every $x \in \mathbb{N}$, so $M \models_\Sigma \varphi$.

Using Structure in First-Order Logic

In first-order logic, models have internal structure:

$$M = (|M|, f^M, R^M, c^M, \dots)$$

This allows statements that talk about elements and relations inside M.

In the institutional view:

- $M \in \mathrm{Mod}(\Sigma)$ abstracts away from internal details.
- Only the satisfaction relation $M \models_\Sigma \varphi$ is required.

This abstraction allows a unified treatment of different logics.

Using Structure in First-Order Logic

Example

Let $\Sigma = \{+, 0, <\}$ and $M = (\mathbb{N}, +, 0, <)$.
We can express:

- **Addition property:**

$$\forall x\, \forall y\, \forall z\, (x < y \rightarrow x + z < y + z)$$

- **Identity law:**

$$\forall x\, (x + 0 = x)$$

- **Existence of successor:**

$$\forall x\, \exists y\, (x < y)$$

Each statement uses the *internal structure* of $(\mathbb{N}, +, 0, <)$: functions $(+)$, constants (0), and relations $(<)$.

Example: Institution of Equational Logic

Equational Logic provides an algebraic institution based on equations between terms.

- Sign – operations and sorts, e.g. $\{+, 0, -\}$ for groups,
- $\mathrm{Sen}(\Sigma)$ – all equations $t_1 = t_2$ built from Σ-terms,
- $\mathrm{Mod}(\Sigma)$ – Σ-algebras interpreting the operations,
- $A \models_\Sigma (t_1 = t_2)$ iff both terms evaluate equally in A.

Idea

Equational logic generalizes algebra: models are algebras, and truth means equality.

Explicit Example

In $A = (\mathbb{Z}, +, -, 0)$:

$$x + 0 = x$$

is true for all $x \in \mathbb{Z}$ since $x + 0 = x$ in integer addition.
Hence $A \models_\Sigma (x + 0 = x)$.

Example: The Real Numbers

The reals $\mathbb{R}$ can be viewed both algebraically and logically.
Let the signature be

$$\Sigma = \{+, 0, <\}.$$

A model for this signature can be written as

$$M = (\mathbb{R}, +^M, <^M, 0^M, 1^M, \cdot^M),$$

where:

- $|M| = \mathbb{R}$ is the carrier set,
- $+^M$ is the usual addition on $\mathbb{R}$,
- $<^M$ is the usual order relation on $\mathbb{R}$,
- $0^M = 0$ is a constant,
- $1^M = 1$ is constant,
- $\cdot^M$ is the usual multiplication on $\mathbb{R}$.

Example: The Real Numbers

- **As an algebra (Equational Logic):**

$$A = (\mathbb{R}, +, \cdot, 0, 1)$$

with equations such as

$$x + 0 = x, \quad x \cdot 1 = x, \quad x + y = y + x.$$

Truth means both sides evaluate equally in A.

- **As a structure (First-Order Logic):**

$$M = (\mathbb{R}, +, \cdot, 0, 1, <)$$

allowing statements like

$$\forall x \, \forall y \, (x < y \rightarrow x + 1 < y + 1).$$

Idea

Adding the relation $<$ turns the algebra into a first-order structure.

Example: Changing Between Models

In an institution, a signature morphism

$$\sigma : \Sigma \to \Sigma'$$

induces a functor

$$\mathrm{Mod}(\sigma) : \mathrm{Mod}(\Sigma') \to \mathrm{Mod}(\Sigma)$$

that *reduces* a model by forgetting structure.

Example (from richer to simpler model)

Let

$$\Sigma = \{+, 0\}, \qquad \Sigma' = \{+, 0, <\},$$

and σ forgets the relation $<$.
Then:

$$\mathrm{Mod}(\Sigma') \ni M' = (\mathbb{R}, +, 0, <) \quad \mapsto \quad \mathrm{Mod}(\sigma)(M') = (\mathbb{R}, +, 0) \in \mathrm{Mod}(\Sigma).$$

Hence the first-order structure $(\mathbb{R}, +, 0, <)$ is *reduced* to its algebraic part $(\mathbb{R}, +, 0)$.

Example: Translating Sentences Between Signatures

In an institution, a signature morphism

$$\sigma : \Sigma \to \Sigma'$$

induces:

$$\mathrm{Sen}(\sigma) : \mathrm{Sen}(\Sigma) \to \mathrm{Sen}(\Sigma'),$$

which *translates sentences* by renaming or extending symbols.

Example: Translating Sentences Between Signatures

Example (formula translation)

Let

$$\Sigma = \{+, 0\}, \qquad \Sigma' = \{+, 0, <\},$$

and define σ as the inclusion map.
Then for a formula

$$\varphi = \forall x\,(x + 0 = x) \in \mathrm{Sen}(\Sigma)$$

we get

$$\mathrm{Sen}(\sigma)(\varphi) = \forall x\,(x + 0 = x) \in \mathrm{Sen}(\Sigma'),$$

the same statement, but now expressed in the richer language that also allows the relation $<$.

Idea

$\mathrm{Sen}(\sigma)$ translates or embeds formulas so that truth remains preserved when the language changes.

Inhalt

Monoidal Category

Definition

A *monoidal category* is a category $\mathcal{C}$ equipped with

- a bifunctor $\otimes : \mathcal{C} \times \mathcal{C} \to \mathcal{C}$,
- a unit object $\mathsf{I} \in \mathcal{C}$,
- natural isomorphisms

$$\alpha_{A,B,C} : (A \otimes B) \otimes C \xrightarrow{\cong} A \otimes (B \otimes C)$$

$$\lambda_A : \mathsf{I} \otimes A \xrightarrow{\cong} A, \quad \rho_A : A \otimes \mathsf{I} \xrightarrow{\cong} A$$

for all $A, B, C \in \mathcal{C}$, satisfying the *pentagon* and *triangle* coherence axioms (Mac Lane's coherence theorem).

Monoidal Category

Pentagon Diagramm

$$\begin{tikzcd}
((A \otimes B) \otimes C) \otimes D \arrow[r, "\alpha_{A,B,C} \otimes \mathrm{id}_D"] \arrow[d, "\alpha_{A \otimes B,C,D}"'] & (A \otimes (B \otimes C)) \otimes D \arrow[r, "\alpha_{A,B \otimes C,D}"] & A \otimes ((B \otimes C) \otimes D) \arrow[d, "\mathrm{id}_A \otimes \alpha_{B,C,D}"] \\
(A \otimes B) \otimes (C \otimes D) \arrow[rr, "\alpha_{A,B,C \otimes D}"'] & & A \otimes (B \otimes (C \otimes D))
\end{tikzcd}$$

Triangle Diagramm

$$\begin{tikzcd}
(A \otimes I) \otimes B \arrow[rr, "\alpha_{A,I,B}"] \arrow[dr, "\rho_A \otimes \mathrm{id}_B"'] & & A \otimes (I \otimes B) \arrow[dl, "\mathrm{id}_A \otimes \lambda_B"] \\
& A \otimes B &
\end{tikzcd}$$

Symmetric Monoidal Category

Definition (Symmetric Monoidal Category)

A *symmetric monoidal category* is a monoidal category $(\mathcal{C}, \otimes, \mathsf{I}, \alpha, \lambda, \rho)$ together with a natural isomorphism (the *symmetry*)

$$s_{A,B} : A \otimes B \xrightarrow{\cong} B \otimes A$$

such that the following diagrams commute for all $A, B, C \in \mathcal{C}$:

Symmetric Monoidal Category

Unit coherence

$$
\begin{tikzcd}
A \otimes \mathbf{I} \arrow[rr, "s_{A,\mathbf{I}}"] \arrow[dr, "\rho_A"'] & & \mathbf{I} \otimes A \arrow[dl, "\lambda_A"] \\
& A &
\end{tikzcd}
$$

Associativity coherence

$$
\begin{tikzcd}
(A \otimes B) \otimes C \arrow[r, "s_{A,B} \otimes \mathrm{id}_C"] \arrow[d, "\alpha_{A,B,C}"'] & (B \otimes A) \otimes C \arrow[r, "\alpha_{B,A,C}"] & B \otimes (A \otimes C) \arrow[d, "\mathrm{id}_B \otimes s_{A,C}"] \\
A \otimes (B \otimes C) \arrow[r, "s_{A,B\otimes C}"'] & (B \otimes C) \otimes A \arrow[r, "\alpha_{B,C,A}"'] & B \otimes (C \otimes A)
\end{tikzcd}
$$

Inverse law

$$
\begin{tikzcd}
A \otimes B \arrow[rr, "s_{A,B}"] \arrow[dr, "\mathrm{id}_{A\otimes B}"'] & & B \otimes A \arrow[dl, "s_{B,A}"] \\
& A \otimes B &
\end{tikzcd}
$$

Example: The Category Vect_k

Vect_k is the category of vector spaces over a field k:

- **Objects:** vector spaces $V, W, \ldots$
- **Morphisms:** k-linear maps $f : V \to W$

Monoidal Structure

- **Tensor product:** $V \otimes W$
- **Unit object:** k (the base field)
- **Associator:**

$$\alpha_{U,V,W} : (U \otimes V) \otimes W \to U \otimes (V \otimes W)$$

- **Unit isomorphisms:** $\lambda_V : k \otimes V \to V, \quad \rho_V : V \otimes k \to V$

Example: The Category Vect_k

Symmetry

$$s_{V,W} : V \otimes W \longrightarrow W \otimes V, \qquad v \otimes w \longmapsto w \otimes v$$

- **Involution:** $s_{W,V} \circ s_{V,W} = \mathrm{id}_{V \otimes W}$
- **Naturality:** $(g \otimes f) \circ s_{V,W} = s_{V',W'} \circ (f \otimes g)$

$(\mathsf{Vect}_k, \otimes, k, \alpha, \lambda, \rho, s)$ is a **symmetric monoidal category**.

Tensoring combines vector spaces, and the swap map shows that the order of tensor factors does not matter up to canonical isomorphism.

Definition (Cartesian Monoidal Category)

Definition (Cartesian Monoidal Category)

A **cartesian monoidal category** is a monoidal category

$$(\mathcal{C}, \times, 1, \alpha, \lambda, \rho)$$

whose monoidal product $\times$ is given by the **categorical product**, and whose unit object is the **terminal object** 1.

Remark:

- The **categorical product** $A \times B$ has projections $\pi_1 : A \times B \to A$, $\pi_2 : A \times B \to B$, and any pair (f, g) factors uniquely through $A \times B$.
- The **terminal object** 1 is an object with exactly *one morphism* $A \to 1$ for every A.

Example: The Category $(\mathrm{Set}, \times, 1)$

Structure

- **Objects:** Sets A, B
- **Morphisms:** Functions $f : A \to B$
- **Monoidal product:** Cartesian product $A \times B$
- **Unit:** Singleton set $1 = \{*\}$ (terminal object)

Canonical maps

- Projections: $\pi_1(a, b) = a$, $\pi_2(a, b) = b$
- Pairing: $\langle f, g \rangle(x) = (f(x), g(x))$

The product $A \times B$ satisfies the universal property, and every set A has a unique map $A \to 1$. Thus $(\mathrm{Set}, \times, 1)$ is a **cartesian monoidal category**.

Co- vs. Cartesian Monoidal Categories

- $(\mathsf{Set}, \times, 1) \to$ **cartesian monoidal** (product of sets, terminal object $1 = \{*\}$)
- $(\mathsf{Vect}_k, \otimes, k) \to$ **monoidal, not cartesian** (no canonical projections $V \otimes W \to V, W$)
- $(\mathsf{Vect}_k, \oplus, 0) \to$ **co-cartesian monoidal** (direct sum, trivial vector space as unit)

Feedback Monoidal Category

Definition (Feedback Monoidal Category)

A **feedback monoidal category** is a **symmetric monoidal category** equipped with:

- an **endofunctor** $F : \mathcal{C} \to \mathcal{C}$,
- and for all $X, Y, S \in \mathcal{C}$ an operation

$$\circlearrowleft_S : \mathrm{Hom}(X \otimes F(S), Y \otimes S) \longrightarrow \mathrm{Hom}(X, Y),$$

  called the **feedback operator**.

This operator must satisfy the following **axioms** for all suitable f, g, h:

- **(Tightening)** $\circlearrowleft_S \left((h \otimes 1_S) \circ f \circ (g \otimes 1_{FS}) \right) = h \circ \circlearrowleft_S (f) \circ g$
- **(Joining)** $\circlearrowleft_{S \otimes T} (f) = \circlearrowleft_S \left(\circlearrowleft_T (f) \right)$
- **(Vanishing)** $\circlearrowleft_I (f) = f$
- **(Strength)** $\circlearrowleft_S (f \otimes g) = g \otimes \circlearrowleft_S (f)$
- **(Sliding)** $\circlearrowleft_T \left((1_Y \otimes g) \circ f \right) = \circlearrowleft_S \left(f \circ (1_X \otimes g) \right)$

Feedback Monoidal Categories

Idea: Feedback monoidal categories describe **processes with feedback loops** in a monoidal setting.

- The operator $\circlearrowleft_S$ connects an output back to an input, forming a **feedback loop**.
- The axioms ensure feedback behaves coherently with composition ($\circ$) and tensor product ($\otimes$).

Remark: If the monoidal structure is **cartesian**, we call it a **feedback cartesian category**. Such categories model stateful systems or causal stream functions.

Cartesian Streams

Definition (Cartesiann Streams)

A **cartesian stream** f : $\chi \to \psi$ consists of two sequences of sets

$$\chi = (X_0, X_1, \dots), \qquad \psi = (Y_0, Y_1, \dots)$$

and, for each $n \geq 0$, a map

$$f_n : X_n \times \cdots \times X_0 \longrightarrow Y_n.$$

For each n, define the cumulative output

$$\hat{f}_n : X_n \times \cdots \times X_0 \longrightarrow Y_n \times \cdots \times Y_0$$

recursively by

$$\hat{f}_0 := f_0, \qquad \hat{f}_{n+1} := (\, f_{n+1} \times \hat{f}_n \,) \circ \left(1_{X_{n+1}} \times \nu_{X_n \times \cdots \times X_0}\right),$$

where $\nu_Z : Z \to Z \times Z$ is the diagonal (duplication) map.

Cartesian Streams

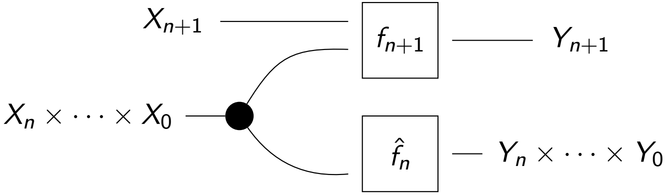
Composition of streams: for $\mathsf{f} : \chi \to \psi$ and $\mathsf{g} : \psi \to \zeta$,

$$(\mathsf{g} \circ \mathsf{f})_n \;:=\; g_n \circ \hat{f}_n.$$

Identity stream: $(1_\chi)_n := \pi_{X_n}$ (the n-th projection).

Connection: Cartesian streams form a category $\mathsf{Stream}_{\mathsf{Set}}$ that models *stateful or causal processes*. This category is a special case of a **feedback cartesian monoidal category**, where the cartesian product serves as the tensor, and the feedback operation corresponds to connecting the output of one step as input to the next.

String Diagramms



Free Category

Definition (Free Category)

Let $G = (V, E, s, t)$ be a directed graph with vertex set V, edge set E, and source and target maps $s, t : E \to V$.
The *free category* on G, denoted $\mathsf{F}(G)$, is defined by:

- $\mathrm{Ob}(\mathsf{F}(G)) = V$;
- $\mathrm{Hom}_{\mathsf{F}(G)}(v, w)$ consists of all finite composable paths $(e_n, \ldots, e_1)$ with $s(e_1) = v$, $t(e_n) = w$, together with the empty path $()$;
- composition is concatenation of paths, and $\mathrm{id}_v = ()$.

The assignment

$$\mathsf{F} : \mathsf{Digraph} \to \mathsf{Cat}, \quad G \mapsto \mathsf{F}(G),$$

is the *free category functor* and Cat is the category of small categories.

Inhalt

Categorical Framework of Explainable AI

- Learning is viewed as an **iterative process with feedback**.
- The process involves a **model/explainer** function that updates internal parameters using feedback from the environment (via an optimizer).
- Formally, an abstract learning agent is a **morphism** in the free feedback Cartesian monoidal category **XLearn**.

Categorical Framework of Explainable AI

XLearn is generated by:

- **Objects:** X, Y, Y^*, P, E
  - Represent input, output, supervision, parameter, and explanation types.
- **Model/Explainer morphism:**

$$\eta : X \times P \to Y \times E$$

  - Produces predictions in Y and explanations in E.
- **Optimizer morphism:**

$$\nabla_Y : Y^* \times Y \times P \to P$$

  - Updates parameters in P.
  - Uses supervisions in Y^*, model predictions in Y, and current parameters in P.

XLearn

Definition (XLearn)

XLearn is the free feedback cartesian category generated by the objects

$$X, Y, Y^*, P, E$$

and by the morphisms

$$\eta : X \times P \to Y \times E \quad \text{and} \quad \nabla_Y : Y^* \times Y \times P \to P.$$

Abstract Learning Agent

Remark: In a feedback cartesian monoidal category, each object X is equipped with morphisms

$$\nu_X : X \to X \times X \quad \text{and} \quad \epsilon_X : X \to e$$

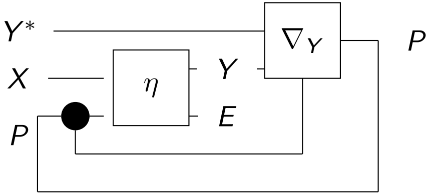
that make it possible to *copy* and *discard* objects.

Definition (Abstract Learning Agent)

An *abstract learning agent* is the morphism in *XLearn* given by:

$$\circlearrowleft_P \left(\nabla_Y \circ (1_{Y^*} \times Y \times \epsilon_E) \circ (1_{Y^*} \times \eta \times 1_P) \circ (1_{Y^*} \times X \times \nu_P)\right).$$

Abstract Learning Agent



Concrete Learning and Explaining Agents

- The free category *XLearn* captures key features of learning agents at an abstract level.
- We can instantiate these agents in *concrete forms* using a **feedback functor** from *XLearn* to the category of Cartesian streams over **Set**, called $Stream_{Set}$.
- This functor maps the abstract structure to concrete settings involving:
  - Various explainers (e.g., decision trees, logistic regression),
  - Different input data types (e.g., images, text),
  - Supervisions, outputs, parameters, and explanations.
- Establishing this mapping requires defining a specific functor called the **translator**.

Agent Translator

Definition (Agent Translator)

An *agent translator* is a feedback cartesian functor

$$\mathcal{T} : \mathsf{XLearn} \to \mathsf{Stream}_{\mathsf{Set}}.$$

Classes of Translators

- Among translators, we distinguish two main classes:
  - **Learning translators:** instantiate *learning agents*.
  - **Explaining translators:** instantiate *explainable learning agents*.
- Intuitively:
  - A **learning agent** is an instance of an abstract learning agent that does *not* provide explanations.
  - A **explaining learning agent** outputs a *non-empty explanation*.

Learning Agent

Definition (Learning Agent)

Given an agent translator $\mathcal{T}$ with

$$\mathcal{T}(E) = \{*\}^{\mathbb{N}},$$

where $\{*\}$ is a singleton set.
A *learning agent (LA)* is the image $\mathcal{T}(\alpha)$, where α is the abstract learning agent.

Explaining Learning Agent

Definition (Explaining Learning Agent)

Let $\mathcal{T}$ be an agent translator, I an institution, and α the abstract learning agent.
The image $\mathcal{T}(\alpha)$ is called

- a *syntactic explaining learning agent* if

$$\mathcal{T}(E) = Sen(\Sigma)^{\mathbb{N}},$$

- and a *semantic explaining learning agent* if

$$\mathcal{T}(E) = Mod(\Sigma)^{\mathbb{N}},$$

for some signature Σ of I.

Remark: Concrete instances of both syntactic and semantic explaining learning agents are referred to as *explaining learning agents (XLA)*.

Inhalt

Simplified Notation

To simplify the notation, we use the following shortcuts:

$$\mathcal{X} = \mathcal{T}(X), \quad \mathcal{Y} = \mathcal{T}(Y), \quad \mathcal{Y}^* = \mathcal{T}(Y^*), \quad \mathcal{P} = \mathcal{T}(P), \quad \mathcal{E} = \mathcal{T}(E),$$

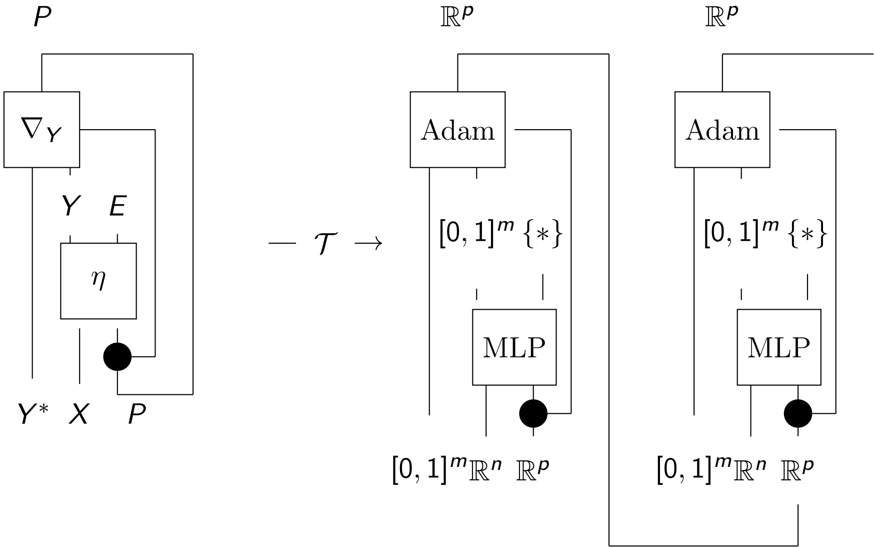
$$\hat{\eta} = \mathcal{T}(\eta), \quad \hat{\nabla}_Y = \mathcal{T}(\nabla_Y), \quad \text{and} \quad \mathcal{Y} = \mathcal{Y}^* \text{ when not stated otherwise.}$$

Multi-Layer Perceptron

A classic multi-layer perceptron (MLP) classifier with an Adam optimizer is an instance of an abstract learning agent whose translator is defined as:

MLP

$$\mathcal{X} = (\mathbb{R}^n)^{\mathbb{N}}, \quad \mathcal{Y} = \mathcal{Y}^* = ([0,1]^m)^{\mathbb{N}}, \quad \hat{\eta}_i = \text{MLP},$$
$$\mathcal{P} = (\mathbb{R}^p)^{\mathbb{N}}, \quad \hat{\nabla}_{\mathcal{Y}_i} = \text{Adam optimizer}, \quad \mathcal{E} = \{*\}^{\mathbb{N}}.$$



P
∇_Y
Y
E
η
Y^*
X
P
$\mathcal{T}$
$\mathbb{R}^p$
Adam
$[0,1]^m$
$\{*\}$
MLP
$[0,1]^m$
$\mathbb{R}^n$
$\mathbb{R}^p$
$\mathbb{R}^p$
Adam
$[0,1]^m$
$\{*\}$
MLP
$[0,1]^m$
$\mathbb{R}^n$
$\mathbb{R}^p$

Multi-Layer Perceptron

In this setting, the MLP is modeled by fixing each morphism component $\hat{\eta}_i = \text{MLP}$, independent of previous inputs. By removing this constraint, we can model broader classes of learning agents such as:

- Recurrent Neural Networks (RNNs),
- Hopfield Networks,
- Transformers.

Neural Architecture Search

A classical Neural Architecture Search (NAS) algorithm is an instance of an abstract learning agent whose translator functor is defined as:

NAS

$$\mathcal{X} = (\mathbb{R}^n)^{\mathbb{N}}, \quad \mathcal{Y} = \mathcal{Y}^* = ([0,1]^m)^{\mathbb{N}}, \quad \hat{\eta}_i = \mathrm{MLP}_i,$$
$$\mathcal{P} = (\mathbb{R}^p)^{\mathbb{N}}, \quad \hat{\nabla}_{\mathcal{Y}_i} = \text{Adam optimizer}, \quad \mathcal{E} = \{*\}^{\mathbb{N}}.$$

Here, each MLP_i represents a different neural architecture at each step.

Explanation

Definition (Explanation)

Given an institution I, an object $\Sigma \in \mathrm{Sign}_I$, and a concrete explainer $\hat{\eta} = \mathcal{T}(\eta) : \mathcal{X} \times \mathcal{P} \to \mathcal{Y} \times \mathcal{E}$, an *explanation* $\mathcal{E} = \mathcal{T}(E)$ in a language Σ is:

- a set of Σ-sentences (*syntactic explanation*), or
- a model of a set of Σ-sentences (*semantic explanation*).

Propositional Logic Explainer

Let I_{PL} be the institution of Propositional Logic and Σ a signature of I_{PL} such that

$$\{x_{flies}, x_{animal}, x_{plane}, x_{dark\_color}, \ldots\} \subseteq \Sigma,$$

with the standard connectives of Boolean Logic $(\neg, \wedge, \vee, \rightarrow)$.

For instance, $\hat{\eta}$ could be an explainer aiming to predict an output in $\mathcal{Y} = \{x_{plane}, x_{bird}, \ldots\}$ given an input in $\mathcal{X}$.

A **syntactic explanation** could be a Σ-sentence such as

$$\varepsilon = x_{flies} \wedge \neg x_{animal} \rightarrow x_{plane},$$

while a **semantic explanation** could be the truth-function of ε.

Isomorphism vs. Equivalence in XAI

Isomorphism (Local Equivalence)

$$\mathrm{Mod}_{\Sigma_2}(\sigma(\varepsilon_1)) \cong \mathrm{Mod}_{\Sigma_2}(\varepsilon_2)$$

Two explanations have identical meaning and structure.

Equivalence of Categories (Global Structural Equivalence)

$$\mathrm{Mod}(\Sigma_1) \simeq \mathrm{Mod}(\Sigma_2)$$

Whole explanation frameworks are structurally similar, translatable via functors preserving truth.

Translators for LA and XLA

We distinguish between two translators:

- $\mathcal{T}$ with $\mathcal{T}(\mu) = \hat{\mu}$ for a LA
- $\mathcal{T}'$ for a XLA

Objects of the latter are denoted with a prime:

$$\mathcal{T}'(Y) = \mathcal{Y}', \quad \mathcal{T}'(E) = \mathcal{E}'.$$

Post-hoc Explainer

Definition (Post-hoc Explainer)

Given a trained Learning Agent (LA) model

$$\hat{\mu} : \mathcal{X} \times \mathcal{P} \to \mathcal{Y},$$

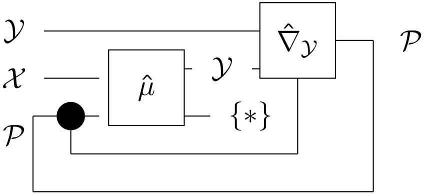
a *post-hoc explainer* is an Explaining Learning Agent (XLA) instantiated by a translator $\mathcal{T}'$ such that

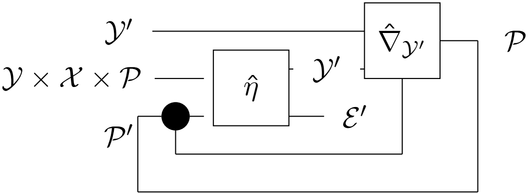
$$\hat{\eta} : \mathcal{X}' \times \mathcal{P}' \to \mathcal{Y}' \times \mathcal{E}', \quad \text{with } \mathcal{X}' = \mathcal{Y} \times \mathcal{X} \times \mathcal{P}.$$

Intuition:

- The explainer $\hat{\eta}$ operates on both inputs and outputs of $\hat{\mu}$.
- It produces an explanation $\mathcal{E}'$ describing the behavior of $\hat{\mu}$.
- Feedback is mediated by ∇'_Y in the XLA framework.

Post-hoc Explainer





Intrinsic Explainer

Definition (Intrinsic Explainer)

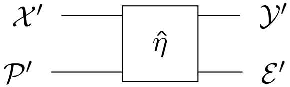
An *intrinsic explainer* is an Explaining Learning Agent (XLA) $\hat{\eta}$ whose input objects are parameters $\mathcal{P}'$ and a set of entries of a database $\mathcal{X}'$:

$$\hat{\eta} : \mathcal{X}' \times \mathcal{P}' \to \mathcal{Y}' \times \mathcal{E}'.$$

Intuition:

- The explanation process is *intrinsic* to the model.
- Explanations $\mathcal{E}'$ are generated directly during training or inference.
- No separate post-hoc model is required (interpretability is built into $\hat{\eta}$).

Intrinsic Explainer



Model-agnostic Explainer

Definition (Model-agnostic Explainer)

Given a Learning Agent (LA) model

$$\hat{\mu} : \mathcal{X} \times \mathcal{P} \to \mathcal{Y},$$

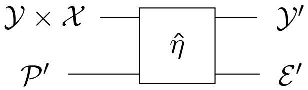
a *model-agnostic explainer* is an Explaining Learning Agent (XLA) $\hat{\eta}$ defined as

$$\hat{\eta} : \mathcal{X}' \times \mathcal{P}' \to \mathcal{Y}' \times \mathcal{E}', \quad \text{with } \mathcal{X}' = \mathcal{Y} \times \mathcal{X}.$$

Intuition:

- The explainer $\hat{\eta}$ operates independently of the internal structure of $\hat{\mu}$.
- It only requires access to inputs $\mathcal{X}$ and outputs $\mathcal{Y}$.
- Examples include LIME and SHAP.

Model-agnostic Explainer



Model-specific Explainer

Definition (Model-specific Explainer)

Given a Learning Agent (LA) model

$$\hat{\mu} : \mathcal{X} \times \mathcal{P} \to \mathcal{Y},$$

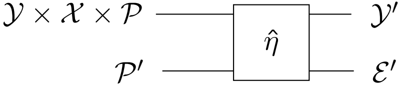
a *model-specific explainer* is an Explaining Learning Agent (XLA) $\hat{\eta}$ that depends on the internal structure or parameters of the model being explained:

$$\hat{\eta} : \mathcal{X}' \times \mathcal{P}' \to \mathcal{Y}' \times \mathcal{E}', \quad \text{with } \mathcal{X}' = \mathcal{Y} \times \mathcal{X} \times \mathcal{P}.$$

Intuition:

- The explainer $\hat{\eta}$ uses the model's parameters or gradients.
- It is *specific* to the structure of $\hat{\mu}$.
- Examples include Grad-CAM or Integrated Gradients.

Model-specific Explainer



Forward-based Explainer

Definition (Forward-based Explainer)

Given a gradient-based Learning Agent (LA) model

$$\hat{\mu} : \mathcal{X} \times \mathcal{P} \to \mathcal{Y},$$

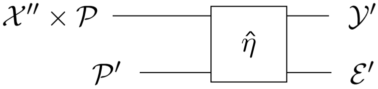
a *forward-based explainer* is an Explaining Learning Agent (XLA) $\hat{\eta}$ defined as

$$\hat{\eta} : \mathcal{X}' \times \mathcal{P}' \to \mathcal{Y}' \times \mathcal{E}', \quad \text{with } \mathcal{X}' = \mathcal{X}'' \times \mathcal{P}.$$

Intuition:

- Operates on the **forward pass** of the LA.
- Uses model activations and parameters to produce explanations.
- Examples: *Activation-based relevance methods*, feature attributions.

Forward-based Explainer



Backward-based Explainer

Definition (Backward-based Explainer)

Given a gradient-based Learning Agent (LA) model

$$\hat{\mu} : \mathcal{X} \times \mathcal{P} \to \mathcal{Y},$$

and an optimizer

$$\nabla_Y : \mathcal{Y} \times \mathcal{Y} \times \mathcal{P} \to \mathcal{P},$$

a *backward-based explainer* is an Explaining Learning Agent (XLA) $\hat{\eta}$ such that

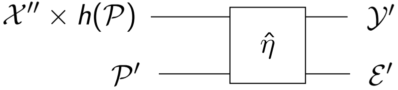
$$\hat{\eta} : \mathcal{X}' \times \mathcal{P}' \to \mathcal{Y}' \times \mathcal{E}', \quad \text{with } \mathcal{X}' = \mathcal{X}'' \times h(\mathcal{P}),$$

where $h(\mathcal{P}) = \frac{\partial \mathcal{L}(\mathcal{Y},\mathcal{Y})}{\partial \mathcal{P}}$ is the gradient of the loss function.

Intuition:

- Operates on the **backward pass** (gradients) of the model.
- Uses gradient information to derive explanations.
- Examples: *Grad-CAM, Integrated Gradients, DeepLIFT*.

Backward-based Explainer



Generalized Learning Schemes

Learning schemes	$\mathcal{T}(X)$	$\mathcal{T}(Y)$	$\mathcal{T}(Y^*)$	$\mathcal{T}(E)$	$\mathcal{T}(\eta)$
Gen. unsup. model	$\diamond^{\mathbb{N}}$	$\diamond^{\mathbb{N}}$	$\{*\}^{\mathbb{N}}$	$\{*\}^{\mathbb{N}}$	$\star$
Gen. sup. model	$\diamond^{\mathbb{N}}$	$\diamond^{\mathbb{N}}$	$\diamond^{\mathbb{N}}$	$\{*\}^{\mathbb{N}}$	$\star$
Gen. continual lear. model	$\diamond^{\mathbb{N}}$	$\diamond^{\mathbb{N}}$	$\diamond^{\mathbb{N}}$	$\{*\}^{\mathbb{N}}$	$\star$
Gen. explaining model	$\diamond^{\mathbb{N}}$	$\diamond^{\mathbb{N}}$	$\diamond^{\mathbb{N}}$	$\diamond^{\mathbb{N}}$	$\star$

Resources

- Barbiero, P., Fioravanti, S., Giannini, F., Tonda, A., Lio, P., Di Lavore, E. (2023). Categorical Foundations of Explainable AI: A Unifying Theory. arXiv. Link
- Goguen, J. A., Burstall, R. M. (1992). Institutions: Abstract Model Theory for Specification and Programming. Link
- Hodges, W. (1997). A Shorter Model Theory. Cambridge University Press. Link
- Nakahira, K. (2023). Diagrammatic category theory. arXiv. Link
- Riehl, E. (2014). Category Theory in Context. (online lecture notes) Link