

# Categorical Foundation of Explainable AI: A Unifying Theory

Pietro Barbiero<sup>\*1</sup>, Stefano Fioravanti<sup>\*2</sup>, Francesco Giannini<sup>\*2</sup>,  
Alberto Tonda<sup>3</sup>, Pietro Lió<sup>1</sup>, Elena Di Lavore<sup>4</sup>

<sup>1</sup>University of Cambridge, UK

<sup>2</sup>Università di Siena, Italy

<sup>3</sup>INRA, Université Paris-Saclay, France

<sup>4</sup>Tallinn University of Technology, Estonia

pb737@cam.ac.uk, stefano.fioravanti@unisi.it, francesco.giannini@unisi.it

## Abstract

Explainable AI (XAI) aims to address the human need for safe and reliable AI systems. However, numerous surveys emphasize the absence of a sound mathematical formalization of key XAI notions—remarkably including the term “*explanation*” which still lacks a precise definition. To bridge this gap, this paper presents the first mathematically rigorous definitions of key XAI notions and processes, using the well-funded formalism of Category theory. We show that our categorical framework allows to: (i) model existing learning schemes and architectures, (ii) formally define the term “*explanation*”, (iii) establish a theoretical basis for XAI taxonomies, and (iv) analyze commonly overlooked aspects of explaining methods. As a consequence, our categorical framework promotes the ethical and secure deployment of AI technologies as it represents a significant step towards a sound theoretical foundation of explainable AI.

## 1 Introduction

Explainable AI (XAI) has emerged as a critical research field to address ethical (Durán and Jongsma 2021; Lo Piano 2020) and legal (Wachter, Mittelstadt, and Russell 2017; EUGDPR 2017) concerns related to the safe deployment of AI technologies. However, several domain surveys remark that key fundamental XAI notions still lack a formal definition, and that the whole field is still missing a unifying and sound formalism (Adadi and Berrada 2018; Palacio et al. 2021).

The notion of *explanation* itself still lacks a proper mathematical formalization, as demonstrated by the attempts to define the concept in the current literature: “*An explanation is an answer to a ‘why?’ question*” (Miller 2019) or “*An explanation is additional meta information, generated by an external algorithm or by the machine learning model itself, to describe the feature importance or relevance of an input instance towards a particular output classification*” (Das and Rad 2020).

We argue that the absence of a rigorous formalization of key explainable AI notions may significantly undermine progress in the field by leading to ill-posed questions, annoying clutter in taxonomies, and narrow perspectives on future research directions. In contrast, a sound mathematical definition would provide a solid foundation for XAI notions and taxonomies, leading to a more organized and efficient research field. This way, researchers could easily iden-

tify knowledge gaps and promising research directions—including the exploration of the (currently overlooked) theoretical side of explainable AI.

**Contributions.** We propose a theoretical framework allowing a unified, comprehensive and rigorous formalization of foundational XAI notions and processes (Section 3). To the best of the authors’ knowledge, this is the first work investigating this research direction in the XAI field. To this objective, we use Category theory as it represents a sound mathematical formalism centered around *processes*, and for this reason, it is widely used in theoretical computer science (Abramsky and Coecke 2004; Selinger 2001; Stein and Staton 2021; Swan et al. 2022; Turi and Plotkin 1997), and, more recently, in AI (Aguinaldo and Regli 2021; Crutwell et al. 2022; Ong and Veličković 2022; Shiebler, Gavranović, and Wilson 2021; Sprunger and Katsumata 2019). In particular, we show that our categorical framework enables us to: model existing learning schemes and architectures (Section 4.1), formally define the term “*explanation*” (Section 4.2), establish a theoretical basis for XAI taxonomies (Section 4.3), and analyze commonly overlooked aspects of explaining methods (Section 4.4).

## 2 Explainable AI Theory: Requirements

In order to build a sound theory, we first need to identify (i) a set of relevant notions, objects, and processes, and (ii) a proper language to formalize them. To formalize objects—such as explanations—we rely on Institution Theory (Goguen and Burstall 1992) as it provides an abstract framework for formal languages’ syntax and semantics (Sec. 2.1). To formalize explainable AI algorithms and their dynamics instead, we rely on Category Theory (Eilenberg and MacLane 1945) as it provides a mathematical framework specifically designed to analyze processes and their dynamics (Sec. 2.2).

### 2.1 Institution Theory: A Framework for Explanations

To provide a formal definition of “*explanation*” (Section 4.2) we rely on institution theory (Goguen and Burstall 1992). Institution theory offers an ideal platform for formalizing explanations—whether expressed through symbolic languages or semantic-based models—as it enables a thorough

analysis of both the structure (syntax) and meaning (semantics) of explanations across diverse languages (Tarski 1944), thus facilitating a deeper understanding of their nature.

More rigorously, an institution  $I$  consists of (i) a category  $\text{Sign}_I$  whose objects are signatures (i.e. vocabularies of symbols), (ii) a functor  $\text{Sen} : \text{Sign}_I \mapsto \text{Set}$  which provides sets of well-formed expressions ( $\Sigma$ -sentences) for each signature  $\Sigma \in \text{Sign}_I^o$ , and (iii) a functor  $\text{Mod} : \text{Sign}_I^{op} \mapsto \text{Set}^1$  that assigns a semantic interpretation (i.e. a world) to the symbols in each signature (Goguen 2005).

A typical example of institution is First-Order Logic (FOL), where the category of signatures is given by sets of relations as objects and arity-preserving functions as morphisms. Sentences and models are defined by standard FOL formulas and structures.

## 2.2 Category Theory: A Framework for XAI Processes

(X)AI processes all share three basic properties: (i) they map (multiple) inputs to (multiple) outputs via a composition of parametric operations (see *monoidal categories*), (ii) they update the parameters of such operations based on some error function (see *feedback categories*), and (iii) they keep updating such parameters over time until convergence (see *cartesian streams*). In the following paragraphs we formalize these (X)AI properties using a categorical language.

**A primer on categories:** Intuitively, a category is a collection of objects and morphisms satisfying specific composition rules.

**Definition 2.1** (Eilenberg and MacLane (1945)). A *category*  $\mathcal{C}$  consists of a class of *objects*  $\mathcal{C}^o$  and, for every  $X, Y \in \mathcal{C}^o$ , a set of *morphisms*  $\text{hom}(X, Y)$  with input type  $X$  and output type  $Y$ . A morphism  $f \in \text{hom}(X, Y)$  is written  $f : X \rightarrow Y$ , and for all morphisms  $f : X \rightarrow Y$  and  $g : Y \rightarrow Z$  there is a *composite* morphism  $f ; g : X \rightarrow Z$ , with composition being associative. For each  $X \in \mathcal{C}^o$  there is an *identity* morphism  $\mathbb{1}_X \in \text{hom}(X, X)$  that makes composition unital.

Well-known examples are the categories  $\text{Set}$ , whose objects are *sets* and morphisms are *functions*, and  $\text{Vec}$ , whose objects are *vector spaces* and morphisms are *linear maps*. Different categories can be connected using special operators called *functors* i.e., mappings of objects and morphisms from one category to another (preserving compositions and identity morphisms). For instance, there is a functor  $\mathcal{F}$  from  $\text{Vec}$  to  $\text{Set}$  that simply ignores the vector space structure.

**Monoidal Categories: compose multi-input/output processes** In this work we are mainly interested in *monoidal categories* as they offer a sound formalism for processes with multiple inputs and outputs (Coecke and Kissinger 2017; Fritz 2020). Monoidal categories (Mac Lane 1978) are categories with additional structure, namely a monoidal product  $\times$  and a neutral element, enabling the composition of morphisms in parallel (cf. Appendix A.1). Notably, monoidal categories allow for a graphical representation of processes using *string diagrams* (Joyal and Street

<sup>1</sup>Given a category  $\mathcal{C}$ ,  $\mathcal{C}^{op}$  denotes its *opposite* category, formed by reversing its morphisms (Mac Lane 1978).

1991). String diagrams enable a more intuitive reasoning over equational theories, and we will use them throughout the paper to provide illustrative, yet formal, definitions of XAI notions. The Coherence Theorem for monoidal categories (Mac Lane 1978) guarantees that string diagrams are a sound and complete syntax for monoidal categories. Thus all coherence equations for monoidal categories correspond to continuous deformations of string diagrams. For instance, given  $f : X \rightarrow Y$  and  $g : Y \rightarrow Z$ , the morphisms  $f ; g : X \rightarrow Z$  and  $\mathbb{1}_X$  are represented as  $X \xrightarrow{f} Y \xrightarrow{g} Z$  and  $X \xrightarrow{\mathbb{1}_X} X$ ; the equation  $f ; \mathbb{1}_Y = f = \mathbb{1}_X ; f$  as

$$X \xrightarrow{f} Y = X \xrightarrow{\mathbb{1}_X} Y = X \xrightarrow{f} Y ;$$

the morphism  $h$  with multiple inputs  $X_1, X_2, X_3$  and outputs  $Y_1, Y_2$  (left), and the parallel composition of two morphisms  $f_1 : X_1 \rightarrow Y_1$  and  $f_2 : X_2 \rightarrow Y_2$  (right) can be represented as follows:



**Feedback Categories: update process states** A common process in machine learning involves the update of the parameters of a function, based on the *feedback* of a loss function. To model this process, we can use *feedback monoidal categories*.

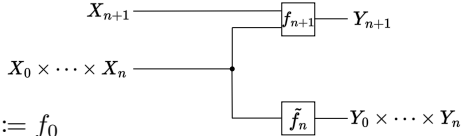
**Definition 2.2** ((Katis, Sabadini, and Walters 2002; Di Loreto et al. 2021)). A *feedback monoidal category* is a symmetric (cf. Appendix A.2) monoidal category  $\mathcal{C}$  endowed with a functor  $\mathcal{F} : \mathcal{C} \rightarrow \mathcal{C}$ , and an operation  $\odot_S : \text{hom}(X \times \mathcal{F}(S), Y \times S) \rightarrow \text{hom}(X, Y)$  for all  $X, Y, S$  in  $\mathcal{C}^o$ , which satisfies a set of axioms (cf. Appendix A.3).

**Cartesian Streams: dynamic update of processes over time** In learning processes, optimizing the loss function often involves a sequence of feedback iterations. Following Sprunger and Katsumata (2019), we use *Cartesian streams* (cf. Appendix A.4) to model this kind of processes. Cartesian streams form a feedback *Cartesian* monoidal category, i.e. are equipped, for every object  $X$ , with morphisms  $\nu_X : X \rightarrow X \times X$  and  $\epsilon_X : X \rightarrow e$  that make it possible to copy and discard objects (cf. Appendix A.2). This makes them the ideal category to formalize (possibly infinite) streams of objects and morphisms.

**Definition 2.3** ((Sprunger and Katsumata 2019; Uustalu and Vene 2008)). Let  $\mathcal{C}$  be a Cartesian category. We call  $\text{Stream}_{\mathcal{C}}$  the category of *Cartesian streams* over  $\mathcal{C}$ , whose objects  $\mathbb{X} = (X_0, X_1, \dots)$  are countable lists of objects in  $\mathcal{C}$ , and given  $\mathbb{X}, \mathbb{Y} \in \text{Stream}_{\mathcal{C}}^o$ , the set of *morphisms*  $\text{hom}(\mathbb{X}, \mathbb{Y})$  is the set of all  $\mathbb{f} : \mathbb{X} \rightarrow \mathbb{Y}$ , where  $\mathbb{f}$  is a family of morphisms in  $\mathcal{C}$ ,  $f_n : X_0 \times \dots \times X_n \rightarrow Y_n$ , for  $n \in \mathbb{N}$ .

In Cartesian streams, a morphism  $f_n$  represents a process that receives a new input  $X_n$  and produces an output  $Y_n$  at time step  $n$ . We can compute the outputs until time  $n$  by combining  $f_0, \dots, f_n$  to get

$\tilde{f}_n: X_0 \times \dots \times X_n \rightarrow Y_0 \times \dots \times Y_n$  as follows:



- $\tilde{f}_0 := f_0$
- $\tilde{f}_{n+1} := (\mathbb{1}_{X_{n+1}} \times \nu_{X_0 \times \dots \times X_n}); (f_{n+1} \times \tilde{f}_n)$

We denote by  $X^{\mathbb{N}}$  the object  $\mathbb{X} \in \text{Stream}_C^o$  such that  $\mathbb{X} = (X, X, \dots)$ , for some  $X \in C^o$ . Notably, Cartesian streams form a feedback monoidal category (Di Lavore, de Felice, and Román 2022) and thus are capable to model dynamic processes with feedback—such as learning processes, as we will show in Theorem 4.1 and 4.2.

### 3 Categorical Framework of Explainable AI

We use feedback monoidal categories and institutions to formalize fundamental (X)AI notions. To this end, we first introduce the definition of “abstract learning agent” (Section 3.1) as a morphism in free feedback monoidal categories, and then we describe a functor instantiating this concept in the concrete feedback monoidal category of  $\text{Stream}_{\text{Set}}$  (Section 3.2). Intuitively, a *free* category serves as a template for a class of categories (e.g., feedback monoids). To generate a free category, we just need to specify a set of objects and morphisms generators. Then we can realize “concrete” instances of a free category  $F$  through a functor from  $F$  to another category  $C$  that preserves the axioms of  $F$  (cf. Appendix A.5).

#### 3.1 Abstract Learning Agents

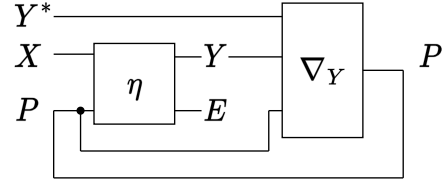
We formalize the abstract concept of an explaining learning agent drawing inspiration from (Cruttwell et al. 2022; Sprunger and Katsumata 2019; Wilson and Zanasi 2021). At a high level, learning can be characterized as an *iterative process with feedback*. This process involves a function (known as *model* or *explainer* in a XAI method) which updates its internal states (e.g., a set of *parameters*) guided by some feedback from the environment (often managed by an *optimizer*). Formally, we define an abstract learning agent as a morphism in the free feedback Cartesian monoidal category  $\text{XLearn}$  generated by:

- the objects  $X, Y, Y^*, P$ , and  $E$  representing input, output, supervision, parameter, and explanation types;
- the *model/explainer* morphism  $\eta: X \times P \rightarrow Y \times E$  which produces predictions in  $Y$  and explanations in  $E$ ;
- the *optimizer* morphism  $\nabla_Y: Y^* \times Y \times P \rightarrow P$  producing updated parameters in  $P$  given supervisions in  $Y^*$ , model/explainer predictions in  $Y$ , and parameters in  $P$ .

**Definition 3.1.**  $\text{XLearn}$  is the free feedback Cartesian category generated by the objects  $X, Y, Y^*, P, E$  and by the morphisms  $\eta: X \times P \rightarrow Y \times E$  and  $\nabla_Y: Y^* \times Y \times P \rightarrow P$ .

The introduction of these morphisms allows us to establish a formal definition of an abstract learning agent.

**Definition 3.2.** An *abstract learning agent* is the morphism in  $\text{XLearn}$  given by the morphisms’ composition:  $\circlearrowleft_P ((\mathbb{1}_{Y^* \times X} \times \nu_P); (\mathbb{1}_{Y^*} \times \eta \times \mathbb{1}_P); (\mathbb{1}_{Y^* \times Y} \times \epsilon_E); \nabla_Y)$



#### 3.2 Concrete Learning and Explaining Agents

The free category  $\text{XLearn}$  allows us to highlight the key features of learning agents from an abstract perspective. However, we can instantiate explaining learning agents in “concrete” forms using a feedback functor from the free category  $\text{XLearn}$  to the category of Cartesian streams over  $\text{Set}$ , i.e.  $\text{Stream}_{\text{Set}}$ . This functor can establish a mapping from our abstract construction to any concrete setting, involving diverse explainers (e.g., decision trees, logistic regression), input data types (e.g., images, text), supervisions, outputs, parameters, or explanations. Achieving this mapping requires the definition of a specific functor we call *translator*.

**Definition 3.3.** An *agent translator* is a feedback Cartesian functor  $\mathcal{T}: \text{XLearn} \rightarrow \text{Stream}_{\text{Set}}$ .

Among translators, we distinguish two significant classes: those that instantiate learning agents and those that instantiate explainable learning agents (Definitions 3.4 and 3.5 respectively). Intuitively, a concrete learning agent is an instance of an abstract learning agent that does not provide any explanation while a concrete explaining learning agent does output an (non-empty) explanation.

**Definition 3.4.** Given an agent translator  $\mathcal{T}$  with  $\mathcal{T}(E) = \{*\}^{\mathbb{N}}$ , where  $\{*\}$  is a singleton set. A *learning agent* (LA) is the image  $\mathcal{T}(\alpha)$ , being  $\alpha$  the abstract learning agent.

The set  $\{*\}^{\mathbb{N}}$  denotes the neutral element of the monoidal product in  $\text{Stream}_{\text{Set}}$  and conveys the absence of explanations. In this case,  $\mathcal{T}(\eta)$  will be simply called *model*, and we will remove the explicit dependence on  $\{*\}$  in the output space as  $T(Y) \times \{*\}^{\mathbb{N}} \cong T(Y)$ . To instantiate explaining learning agents instead, we introduce two distinct types of translators: the semantic and the syntactic translator. This choice is motivated by the foundational elements of institution theory, namely sentences and models: Sentences correspond to well-formed *syntactic* expressions, while models capture the *semantic* interpretations of these sentences (Goguen and Burstall 1992). We refer to concrete instances of both syntactic and semantic explaining learning agents as “explaining learning agents” (XLA).

**Definition 3.5.** Let  $\mathcal{T}$  be an agent translator,  $I$  an institution and  $\alpha$  the abstract learning agent. The image  $\mathcal{T}(\alpha)$  is said a *syntactic explaining learning agent* if  $\mathcal{T}(E) = \text{Sen}(\Sigma)^{\mathbb{N}}$  and a *semantic explaining learning agent* if  $\mathcal{T}(E) = \text{Mod}(\Sigma)^{\mathbb{N}}$ , for some signature  $\Sigma$  of  $I$ .

The high degree of generality in this formalization enables the definition of any real-world learning setting and learning agent (to the author knowledge). Indeed, using Cartesian streams as the co-domain of translators, we can effectively model a wide range of learning use cases, including (but not limited to) those involving iterative feedback.

To simplify the notation, in the following sections we use the shortcuts:  $\mathcal{X} = \mathcal{T}(X)$ ,  $\mathcal{Y} = \mathcal{T}(Y)$ ,  $\mathcal{Y}^* = \mathcal{T}(Y^*)$ ,  $\mathcal{P} = \mathcal{T}(P)$ ,  $\mathcal{E} = \mathcal{T}(E)$ ,  $\hat{\eta} = \mathcal{T}(\eta)$ ,  $\hat{\nabla}_{\mathcal{Y}} = \mathcal{T}(\nabla_Y)$ , and  $\mathcal{Y} = \mathcal{Y}^*$  when not specified.

## 4 Impact on XAI and Key Findings

### 4.1 Finding #1: Our framework models existing learning schemes and architectures

As a proof of concept, in the following examples we show how the proposed categorical framework makes it possible to capture the structure of some popular learning algorithms such as neural networks.

**Example 4.1.** A classic multi-layer perceptron (MLP, (Rumelhart, Hinton, and Williams 1985)) classifier with Adam optimizer (Kingma and Ba 2014) is an instance of an abstract learning agent whose translator functor is defined as follows (Section 4.1):  $\mathcal{X} = (\mathbb{R}^n)^{\mathbb{N}}$ ,  $\mathcal{Y} = \mathcal{Y}^* = ([0, 1]^m)^{\mathbb{N}}$ ,  $\hat{\eta}_i$  being an MLP for all  $i$ ,  $\mathcal{P}$  the space of the MLP parameters, e.g.  $\mathcal{P} = (\mathbb{R}^p)^{\mathbb{N}}$ ,  $\hat{\nabla}_{\mathcal{Y}_i}$  being e.g. the Adam optimizer, and  $\mathcal{E} = \{*\}^{\mathbb{N}}$ .

In Example 4.1 we successfully model an MLP by constraining the components of the morphism  $\eta$  to have constant values  $\hat{\eta}_i = \text{MLP}$  (independent of the first  $i - 1$  inputs). However, by removing this constraint, we can instantiate a broader class of learning agents including e.g. recurrent neural networks (Hochreiter and Schmidhuber 1997; Hopfield 1982) and transformers (Vaswani et al. 2017). In these scenarios, the neural functions become dependent on previous inputs, effectively capturing the input stream. Additionally, we can also model learning settings where a model’s architecture changes over time, as in neural architecture search (Elsken, Metzen, and Hutter 2019).

**Example 4.2.** A classical Neural Architecture Search algorithm (Elsken, Metzen, and Hutter 2019) is an instance of an abstract learning agent whose translator functor is defined as follows:  $\mathcal{X} = (\mathbb{R}^n)^{\mathbb{N}}$ ,  $\mathcal{Y} = \mathcal{Y}^* = ([0, 1]^m)^{\mathbb{N}}$ ,  $\hat{\eta}_i = \text{MLP}_i$  being a different neural architecture for every step,  $\mathcal{P}$  the space of the MLPs parameters, e.g.  $\mathcal{P} = (\mathbb{R}^p)^{\mathbb{N}}$ ,  $\hat{\nabla}_{\mathcal{Y}_i}$  being the Adam optimizer, and  $\mathcal{E} = \{*\}^{\mathbb{N}}$ .

To the best of our knowledge, the proposed formalism is general enough to potentially encompass any known learning process providing or not providing explanations. Indeed, the objects  $X, Y, Y^*, E$  can be instantiated through a suitable translator functor to have the desired characteristics, e.g.  $\mathcal{T}(Y)$  could include the explanations space, making the explanations optimizable, or  $\mathcal{T}(Y^*) = \{*\}^{\mathbb{N}}$ , giving an unsupervised model. Further examples showing the flexibility of our framework can be found in the Appendix A.7.

### 4.2 Finding #2: Our framework enables a formal definition of “explanation”

Our theoretical framework allows us to provide the first formal definition of the term “explanation”, which embodies the very essence and purpose of explainable AI. Our formalization goes beyond a mere definition of “explanation”, as it highlights a natural distinction between different forms

of explanations. Indeed, institution theory allows a straightforward characterization of syntactic and semantic explanations. While both forms of explanations are prevalent in the current XAI literature, their distinctions are often overlooked, thus limiting a deep understanding of the true nature and intrinsic limitations of a given explanation.

**Definition 4.3.** Given an institution  $I$ , an object  $\Sigma$  of  $\text{Sign}_I$ , and a concrete explainer  $\hat{\eta} = \mathcal{T}(\eta) : \mathcal{X} \times \mathcal{P} \rightarrow \mathcal{Y} \times \mathcal{E}$ , an *explanation*  $\mathcal{E} = \mathcal{T}(E)$  in a language  $\Sigma$  is a set of  $\Sigma$ -sentences (*syntactic explanation*) or a model of a set of  $\Sigma$ -sentences (*semantic explanation*).

We immediately follow up our definition with a concrete example to make it more tangible.

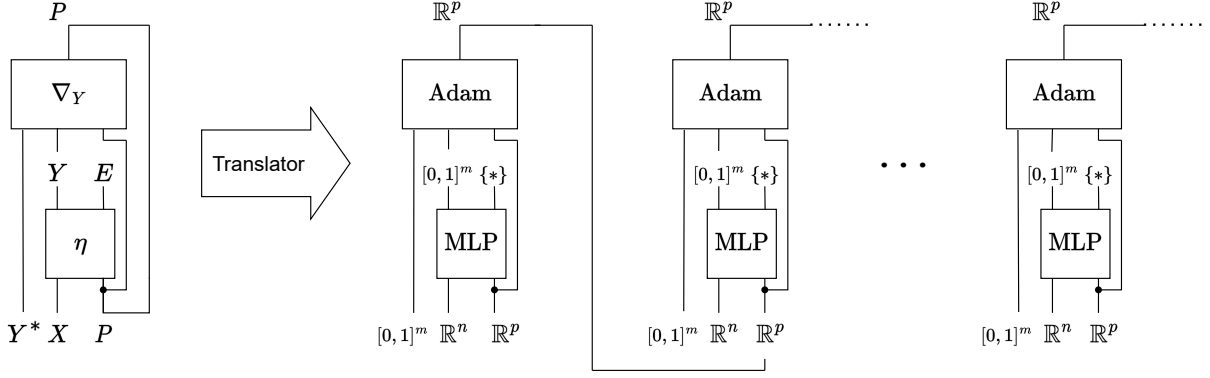
**Example 4.4.** Let  $I_{PL}$  be the institution of Propositional Logic and  $\Sigma$  a signature of  $I_{PL}$  such that  $\{x_{flies}, x_{animal}, x_{plane}, x_{dark\_color}, \dots\} \subseteq \Sigma$  and with the standard connectives of Boolean Logic, i.e.  $\neg, \wedge, \vee, \rightarrow$ . For instance,  $\hat{\eta}$  could be an explainer aiming at predicting an output in  $\mathcal{Y} = \{x_{plane}, x_{bird}, \dots\}$  given an input in  $\mathcal{X}$ . Then a syntactic explanation could consist of a  $\Sigma$ -sentence like  $\varepsilon = x_{flies} \wedge \neg x_{animal} \rightarrow x_{plane}$ , a semantic explanation could be the truth-function of  $\varepsilon$ .

Our definition of explanation not only generalizes and formalizes existing definitions in the literature but also encompasses key aspects discussed by different researchers. For instance, according to Miller (2019), an explanation is “an answer to a ‘why?’ question”; Das and Rad (2020) define explanation as “additional meta information, generated by an external algorithm or by the machine learning model itself, to describe the feature importance or relevance of an input instance towards a particular output classification”; while Palacio et al. (2021) describe explanation as “the process of describing one or more facts, such that it facilitates the understanding of aspects related to said facts”. Our definition of explanation incorporates all these notions, as  $\Sigma$ -sentences and their models can provide any form of statement related to the explaining learning agent and its inputs/outputs. This encompasses additional meta information and feature relevance (Das and Rad 2020), description of facts related to a learning process (Palacio et al. 2021), or insights into why a specific output is obtained from a given input (Miller 2019).

Furthermore, Tarski (1944) and Goguen and Burstall (1992) proved how the semantics of “truth” is invariant under change of signature. This means that we can safely use signature morphisms to switch from one “notation” to another, inducing consistent syntactic changes in a  $\Sigma$ -sentence without conditioning the “meaning” or the “conclusion” of the sentence (Goguen and Burstall 1992). As a result, signature morphisms can translate a certain explanation between different signatures, hence paving the way to study “communication” as well as “understanding” between XLAs.

### 4.3 Finding #3: Our framework provides a theoretical foundation for XAI taxonomies

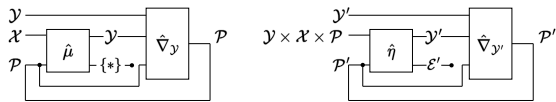
Using our categorical constructions, we can develop a theory-grounded taxonomy of XAI methods that goes beyond current ones and catalogues existing approaches in



a systematic and rigorous manner. We recognize the importance of such a foundation due to the ongoing debates and challenges faced by current taxonomies in comprehensively encompassing all existing XAI methods. Indeed, existing approaches in the XAI literature have only provided subjective viewpoints on the field, distinguishing methods according to controversial criteria such as: the scope/methodology/usage of explanations (Das and Rad 2020); the how/what/why of explanations (Palacio et al. 2021); the relationship between explainers and systems being explained, e.g., intrinsic/post-hoc, model-specific/agnostic, local/global (Molnar 2020); or the specific data types, such as images (Kulkarni, Shivananda, and Sharma 2022), graphs (Li et al. 2022), natural language (Danilevsky et al. 2020), and tabular data (Di Martino and Delmastro 2022). However, these taxonomies lack a solid and grounded motivation and rely primarily on subjective preferences. As a result, they are unable to draw universal conclusions and provide a general understanding of the field. On the contrary, our taxonomy aims to fill this gap by providing a comprehensive classification of XAI methods grounded in our theoretical framework. As an example, we use the proposed categorical framework to explicitly describe the following macro-categories of XAI methods (Das and Rad 2020; Molnar 2020; Palacio et al. 2021), where we distinguish between an LA instantiated by a translator  $\mathcal{T}$ , with  $\mathcal{T}(\eta) = \hat{\mu}$ , and an XLA instantiated by  $\mathcal{T}'$ . We will refer to the objects of the latter using prime, e.g.  $\mathcal{T}'(Y) = \mathcal{Y}'$ .

**Post-hoc and Intrinsic.** XAI surveys currently distinguish between intrinsic and post-hoc explainers. Informally, the key difference is that intrinsic XAI methods evolve model parameters and explanations at the same time, whereas post-hoc methods extract explanations from pre-trained models (Das and Rad 2020; Molnar 2020).

**Post-hoc explainer.** Given a trained LA model  $\hat{\mu} : \mathcal{X} \times \mathcal{P} \rightarrow \mathcal{Y}$ , a post-hoc explainer is an XLA explainer instantiated by  $\mathcal{T}'$  such that  $\hat{\eta} : \mathcal{X}' \times \mathcal{P}' \rightarrow \mathcal{Y}' \times \mathcal{E}'$ , with  $\mathcal{X}' = \mathcal{Y} \times \mathcal{X} \times \mathcal{P}$ :



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

$$\begin{array}{c} \mathcal{X}' \\ \mathcal{P}' \end{array} \rightarrow \hat{\eta} \rightarrow \begin{array}{c} \mathcal{Y}' \\ \mathcal{E}' \end{array}$$

Common intrinsic explainers are logic/rule-based (Barbiero et al. 2023; Breiman et al. 1984; Ciravegna et al. 2023; Friedman and Popescu 2008; Manhaeve et al. 2018; Schmidt and Lipson 2009; Yang, Rudin, and Seltzer 2017), linear (Doshi-Velez, Wallace, and Adams 2015; Hastie 2017; Nelder and Wedderburn 1972; Santosa and Symes 1986; Tibshirani 1996; Verhulst 1845), and prototype-based (Fix and Hodges 1989; Kaufmann 1987) approaches; while well-known post-hoc explainers include saliency maps (Selvaraju et al. 2017; Simonyan, Vedaldi, and Zisserman 2013), surrogate models (Lundberg and Lee 2017; Ribeiro, Singh, and Guestrin 2016a, 2018), and concept-based approaches (Espinosa Zarlenga et al. 2022; Ghorbani, Abid, and Zou 2019; Ghorbani et al. 2019; Kim, Khanna, and Koyejo 2016).

**Model-Agnostic and Model-Specific** Intuitively, model-agnostic explainers extract explanations independently from the architecture and/or parameters of the model being explained, whereas model-specific explainers depend on the architecture and/or parameters of the model.

**Model-agnostic explainer.** Given an LA model  $\hat{\mu} : \mathcal{X} \times \mathcal{P} \rightarrow \mathcal{Y}$ , a model-agnostic explainer is an XLA explainer  $\hat{\eta} : \mathcal{X}' \times \mathcal{P}' \rightarrow \mathcal{Y}' \times \mathcal{E}'$ , such that  $\mathcal{X}' = \mathcal{Y} \times \mathcal{X}$ .

$$\begin{array}{c} \mathcal{Y} \times \mathcal{X} \\ \mathcal{P}' \end{array} \rightarrow \hat{\eta} \rightarrow \begin{array}{c} \mathcal{Y}' \\ \mathcal{E}' \end{array}$$

**Model-specific explainer.** A model-specific explainer simply differentiates from a model-agnostic, as the XLA explainer  $\hat{\eta} : \mathcal{X}' \times \mathcal{P}' \rightarrow \mathcal{Y}' \times \mathcal{E}'$  has  $\mathcal{X}' = \mathcal{Y} \times \mathcal{X} \times \mathcal{P}$ . Typical examples of model-agnostic explainers include sur-

$$\begin{array}{c} \mathcal{Y} \times \mathcal{X} \times \mathcal{P} \\ \mathcal{P}' \end{array} \rightarrow \hat{\eta} \rightarrow \begin{array}{c} \mathcal{Y}' \\ \mathcal{E}' \end{array}$$

rogate models (Lundberg and Lee 2017; Ribeiro, Singh,

and Guestrin 2016a, 2018) and some concept-based approaches (Ghorbani, Abid, and Zou 2019; Ghorbani et al. 2019; Kim, Khanna, and Koyejo 2016). Among renowned model-specific explainers instead we can include all model-intrinsic explainers (Molnar 2020) and some post-hoc explainers such as saliency maps (Selvaraju et al. 2017; Simonyan, Vedaldi, and Zisserman 2013) as they can only explain gradient-based systems.

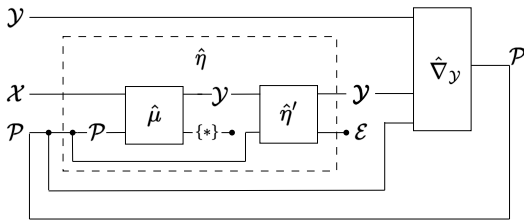
**Forward and Backward.** Another relevant difference among XAI methods for gradient-based models is whether the explainer relies on the upcoming parameters (Petsiuk, Das, and Saenko 2018; Zeiler and Fergus 2014; Zintgraf et al. 2017) or the gradient of the loss on the parameters in the learning optimizer (Selvaraju et al. 2017; Simonyan, Vedaldi, and Zisserman 2013).

**Forward-based explainer.** Given a gradient-based LA model  $\hat{\mu} : \mathcal{X} \times \mathcal{P} \rightarrow \mathcal{Y}$ , a forward-based explainer is an XLA explainer  $\hat{\eta} : \mathcal{X}' \times \mathcal{P}' \rightarrow \mathcal{Y}' \times \mathcal{E}'$  with  $\mathcal{X}' = \mathcal{X}'' \times \mathcal{P}$ .

$$\begin{array}{c} \mathcal{X}'' \times \mathcal{P} \\ \mathcal{P}' \text{---} \boxed{\hat{\eta}} \text{---} \mathcal{Y}' \\ \mathcal{E}' \end{array}$$

**Backward-based explainer.** Given a gradient-based LA model  $\hat{\mu} : \mathcal{X} \times \mathcal{P} \rightarrow \mathcal{Y}$  and an optimizer  $\hat{\nabla}_{\mathcal{Y}} : \mathcal{Y} \times \mathcal{Y} \times \mathcal{P} \rightarrow \mathcal{P}$ , a backward-based explainer is an XLA explainer  $\hat{\eta} : \mathcal{X}' \times \mathcal{P}' \rightarrow \mathcal{Y}' \times \mathcal{E}'$  where,  $\mathcal{X}' = \mathcal{X}'' \times h(\mathcal{P})$ , being  $h(\mathcal{P}) = \frac{\partial \mathcal{L}(\mathcal{Y}, \mathcal{Y})}{\partial \mathcal{P}}$  the gradient of the loss function  $\mathcal{L}$  on  $\mathcal{P}$ .

**A case study: Concept bottleneck models.** Concept bottleneck models (CBM) (Koh et al. 2020) are recent XAI architectures which first predict a set of human-understandable objects called “concepts” and then use these concepts to solve downstream classification tasks. Our framework allows to formally define even these advanced XAI structures as follows: A CBM is an XLA such that  $\hat{\eta}$  is composed of a concept predictor  $\hat{\mu} : \mathcal{X} \times \mathcal{P} \rightarrow \mathcal{Y}$  and a task predictor  $\hat{\eta}' : \mathcal{Y} \times \mathcal{P} \rightarrow \mathcal{Y} \times \mathcal{E}$ ,  $\mathcal{Y}$  is the set of classes and  $\mathcal{P} = \mathcal{P}' \times \mathcal{P}''$  is the product of the parameter space of the two models.



Overall these examples give a taste of the flexibility and expressive power of our categorical framework demonstrating how it can successfully encompass existing XAI approaches and taxonomies.

#### 4.4 Finding #4: Our framework emphasizes commonly overlooked aspects of explanations

Current XAI taxonomies often neglect the distinction between syntactic and semantic approaches. On the contrary, our Definition 3.5 provides a natural distinction between these two forms of XLAs, thus introducing a novel perspective to analyze XAI methods. On the one hand, syn-



Figure 1: Saliency map.

tactic approaches work on the structure of symbolic languages and operate on  $\Sigma$ -sentences. Notable examples of syntactic approaches are proof systems such as natural deduction (Prawitz 2006) and sequent calculi (Takeuti 2013) which are designed to operate on formal languages such as first-order logic. On the other hand, semantic approaches provide explanations related to the meaning or interpretation of sentences as a model of a language. Most XAI techniques actually fall into this class of methods as semantic explanations establish a direct connection with specific problem domains (Karasmanoglou, Antonakakis, and Zervakis 2022; Lundberg and Lee 2017; Ribeiro, Singh, and Guestrin 2016a; Simonyan, Vedaldi, and Zisserman 2013). The following examples show the relation between a syntactic and a semantic technique emphasizing connections between XAI methods that often slip unnoticed.

**Example 4.5.** The Gradient-weighted Class Activation Mapping (Grad-CAM), (Simonyan, Vedaldi, and Zisserman 2013) is a classic example of a semantic (backward) XLA, whose institution can be defined as a fragment of FOL with all the signatures’ objects consisting of a single predicate and a finite set of constants. Intuitively, in a classification task the constants represent the pixels of an image and the relation represents the saliency degree of each pixel for the class prediction. Sentences and models are defined as in FOL by using the provided signature. A general syntactic explanation in this institution can be easily expressed by taking a signature  $\Sigma = \{S, p_1, p_2, \dots\}$  with  $S$  the unary saliency predicate and  $p_i$  the constant for the  $i$ -th pixel. Assuming to collect the most “salient” pixels in the set  $SalPix$ , the syntactic explanation may be expressed by:  $\bigwedge_{p \in SalPix} S(p)$ . Figure 1 instead represents a semantic explanation where Grad-CAM interprets predicates and constants in the syntactic formula assigning them a meaning (i.e., concrete values).

**Example 4.6.** Another classic example of semantic XLAs are feature importance methods (Wei, Lu, and Song 2015), such as LIME, (Ribeiro, Singh, and Guestrin 2016b). As saliency maps, LIME relies on the institution  $I_{un}$ , as their signatures consist on a set of constants  $f_i$  (each for every considered feature) and a single predicate  $R$  to express their relevance. Syntactic explanations have the form  $\bigwedge_{f \in ImpFeat} R(f)$  where the set  $ImpFeat$  collects the most “relevant” features for a task. Figure 2 shows an example of a semantic explanation of LIME, where the length of each

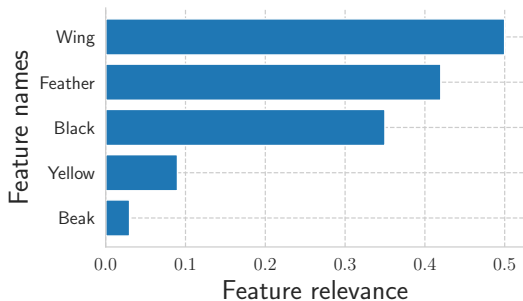


Figure 2: Feature importance.

bar represents the relevance of the corresponding feature.

**Comparing the expressive power of explanations.** The Grad-CAM and LIME examples offer the ideal setting to show how our framework can formally assess the expressive power of different forms of explanations, drawing connections between apparently different XLAs. Indeed, provided that both Grad-CAM and LIME signatures contain the same number of pixels/features, these two forms of explanations syntactically possess the same expressive power (while differing in their semantic models and generating algorithms for the explanations). Consequently, we can convert any saliency map into an equivalent feature importance representation and vice versa (using an arity-preserving function), without compromising their meaning/truth value (Tarski 1944). This is generally true, whenever it is possible to define an arity-preserving mapping between different signatures in the same institution. This observation underscores an often overlooked aspect in XAI literature: evaluating the expressive power of explanations requires comparing their signatures and syntax, not the way these explanations are visualized. User studies that compare different forms of visualization essentially assess the visualization’s expressive power, which relates to human understanding, rather than the expressive power of an explanation itself.

**Limitations of semantic explanations.** The connection between a semantic explanation and a specific context (e.g., an image) may stimulate human imagination, but it often limits the scope and robustness of the explanation, hindering human understanding in the long run (Ghorbani, Abid, and Zou 2019; Rudin 2019). On the contrary, symbolic languages such as first-order logic or natural language are preferable for conveying meaningful messages as explanations as suggested by Kahneman (2011); Marcus (2020). For this reason, a promising but often overlooked research direction consists in accurately lifting semantic explanations into a symbolic language in order to provide a syntactic explanation (Breiman et al. 1984; Letham et al. 2015; Costa et al. 2018; Guidotti et al. 2018; Ribeiro, Singh, and Guestrin 2018; Ciravegna et al. 2023). By recognizing the importance of this distinction, our formalization can provide a suitable basis to gaining deeper insights into the limitations of different forms of explanations.

## 5 Discussion

**Significance and relations with other XAI foundational works.** The explainable AI research field is growing at a considerable rate (Minh et al. 2022) and it now has a concrete impact on other research disciplines (Cranmer et al. 2019; Davies et al. 2021; Jiménez-Luna, Grisoni, and Schneider 2020) as well as on the deployment of AI technologies (Durán and Jongsma 2021; Lo Piano 2020; Wachter, Mittelstadt, and Russell 2017). This rising interest in explainable AI increases the need for a sound foundation and taxonomy of the field (Adadi and Berrada 2018; Palacio et al. 2021). Indeed, existing reviews and taxonomies significantly contribute in: (i) describing and clustering the key methods and trends in the XAI literature (Molnar 2020), (ii) proposing the first qualitative definitions for the core XAI terminology (Das and Rad 2020), (iii) relating XAI methodologies, aims, and terminology with other scientific disciplines (Miller 2019), and (iv) identifying the key knowledge gaps and research opportunities (Das and Rad 2020). However, most of these works acknowledge the need for a more sound mathematical foundation and formalism to support the field. Our framework arises to fill this gap. In particular our methodology formalizes key XAI notions for the first time, using the category of Cartesian streams and the category of signatures. Our work also draws from Sprunger and Katsumata (2019) who propose Cartesian streams to model gradient-based learning, and Cruttwell et al. (2022) who model gradient-based learning using the category of lenses (Riley 2018). The categorical formalisms of lenses and streams are closely related (Di Lavore, de Felice, and Román 2022). Intuitively, lenses can be used to encode one-stage processes, while streams can encode processes with time indexed by natural numbers, i.e. providing a more suitable description of the dynamic process of learning. However, our work opens to more general AI systems which are not necessarily gradient-based by generalizing the category of lenses with Cartesian streams.

**Limitations** In this work we face the challenging task of formalizing (previously informal) notions such as “explanation”, while acknowledging the ongoing debate over their meaning, not only within the AI community but also in philosophy, epistemology, and psychology. Our formalization offers a robust theory-grounded foundation for explainable AI, but it does require readers to engage with abstract categorical structures. However, embracing this initial challenge brings a substantial payoff by enabling us to achieve a comprehensive and unified theory of the field, encompassing all the pertinent instantiations of XAI notions, structures, explanations, and paradigms.

**Conclusion** This work presents the first formal theory of explainable AI. In particular, we formalized key notions and processes that were still lacking a rigorous definition. We then show that our categorical framework enables us to: (i) model existing learning schemes and architectures, (ii) formally define the term “explanation”, (iii) establish a theoretical basis for XAI taxonomies, and (iv) emphasize commonly overlooked aspects of XAI methods, like the comparison between syntactic and semantics explanations. Through

this work, we provide a first answer to the pressing need for a sound foundation and formalism in XAI as advocated by the current literature (Adadi and Berrada 2018; Palacio et al. 2021). While our taxonomy provides guidance to navigate the field, our formalism strengthens the reputation of explainable AI encouraging a safe and ethical deployment of AI technologies. We think that this work may contribute in paving the way for new research directions in XAI, including the exploration of previously overlooked theoretical side of explainable AI, and the mathematical definition of other foundational XAI notions, like “understandability” and “trustworthiness”.

## References

- Abramsky, S.; and Coecke, B. 2004. A categorical semantics of quantum protocols. *Proceedings of the 19th Annual IEEE Symposium on Logic in Computer Science, 2004.*, 415–425.
- Adadi, A.; and Berrada, M. 2018. Peeking inside the black-box: a survey on explainable artificial intelligence (XAI). *IEEE access*, 6: 52138–52160.
- Aguinaldo, A.; and Regli, W. 2021. A Graphical Model-Based Representation for Classical AI Plans using Category Theory. In *ICAPS 2021 Workshop on Explainable AI Planning*.
- Barbiero, P.; Ciravegna, G.; Giannini, F.; Zarlenga, M. E.; Magister, L. C.; Tonda, A.; Lio, P.; Precioso, F.; Jamnik, M.; and Marra, G. 2023. Interpretable Neural-Symbolic Concept Reasoning. *arXiv preprint arXiv:2304.14068*.
- Breiman, L.; Friedman, J.; Stone, C. J.; and Olshen, R. A. 1984. *Classification and regression trees*. CRC press.
- Ciravegna, G.; Barbiero, P.; Giannini, F.; Gori, M.; Lió, P.; Maggini, M.; and Melacci, S. 2023. Logic explained networks. *Artificial Intelligence*, 314: 103822.
- Coecke, B.; and Kissinger, A. 2017. *Picturing Quantum Processes - A first course in Quantum Theory and Diagrammatic Reasoning*. Cambridge University Press.
- Costa, F.; Ouyang, S.; Dolog, P.; and Lawlor, A. 2018. Automatic generation of natural language explanations. In *Proceedings of the 23rd international conference on intelligent user interfaces companion*, 1–2.
- Cranmer, M. D.; Xu, R.; Battaglia, P.; and Ho, S. 2019. Learning symbolic physics with graph networks. *arXiv preprint arXiv:1909.05862*.
- Cruttwell, G. S.; Gavranović, B.; Ghani, N.; Wilson, P.; and Zanasi, F. 2022. Categorical foundations of gradient-based learning. In *European Symposium on Programming*, 1–28. Springer, Cham.
- Danilevsky, M.; Qian, K.; Aharonov, R.; Katsis, Y.; Kawas, B.; and Sen, P. 2020. A survey of the state of explainable AI for natural language processing. *arXiv preprint arXiv:2010.00711*.
- Das, A.; and Rad, P. 2020. Opportunities and Challenges in Explainable Artificial Intelligence (XAI): A Survey. *ArXiv*, abs/2006.11371.
- Davies, A.; Veličković, P.; Buesing, L.; Blackwell, S.; Zheng, D.; Tomašev, N.; Tanburn, R.; Battaglia, P.; Blundell, C.; Juhász, A.; et al. 2021. Advancing mathematics by guiding human intuition with AI. *Nature*, 600(7887): 70–74.
- Di Lavore, E.; de Felice, G.; and Román, M. 2022. Monoidal Streams for Dataflow Programming. In *Proceedings of the 37th Annual ACM/IEEE Symposium on Logic in Computer Science*. New York, NY, USA: Association for Computing Machinery. ISBN 9781450393515.
- Di Lavore, E.; Gianola, A.; Román, M.; Sabadini, N.; and Sobociński, P. 2021. A canonical algebra of open transition systems. In *Formal Aspects of Component Software: 17th International Conference, FACS 2021, Virtual Event, October 28–29, 2021, Proceedings 17*, 63–81. Springer.
- Di Martino, F.; and Delmastro, F. 2022. Explainable AI for clinical and remote health applications: a survey on tabular and time series data. *Artificial Intelligence Review*, 1–55.
- Doshi-Velez, F.; Wallace, B. C.; and Adams, R. 2015. Graph-sparse lda: a topic model with structured sparsity. In *Twenty-Ninth AAAI conference on artificial intelligence*.
- Durán, J. M.; and Jongsma, K. R. 2021. Who is afraid of black box algorithms? On the epistemological and ethical basis of trust in medical AI. *Journal of Medical Ethics*, 47(5): 329–335.
- Eilenberg, S.; and MacLane, S. 1945. General theory of natural equivalences. *Transactions of the American Mathematical Society*, 58(2): 231–294.
- Elsken, T.; Metzen, J. H.; and Hutter, F. 2019. Neural architecture search: A survey. *The Journal of Machine Learning Research*, 20(1): 1997–2017.
- Espinosa Zarlenga, M.; Barbiero, P.; Ciravegna, G.; Marra, G.; Giannini, F.; Diligenti, M.; Shams, Z.; Precioso, F.; Melacci, S.; Weller, A.; et al. 2022. Concept Embedding Models: Beyond the Accuracy-Explainability Trade-Off. *Advances in Neural Information Processing Systems*, 35: 21400–21413.
- EUGDPR. 2017. GDPR. General data protection regulation.
- Fix, E.; and Hodges, J. L. 1989. Discriminatory analysis. Nonparametric discrimination: Consistency properties. *International Statistical Review/Revue Internationale de Statistique*, 57(3): 238–247.
- Fox, T. 1976. Coalgebras and cartesian categories. *Communications in Algebra*, 4(7): 665–667.
- Friedman, J. H.; and Popescu, B. E. 2008. Predictive learning via rule ensembles. *The annals of applied statistics*, 916–954.
- Fritz, T. 2020. A synthetic approach to Markov kernels, conditional independence and theorems on sufficient statistics. *Advances in Mathematics*, 370: 107239.
- Ghorbani, A.; Abid, A.; and Zou, J. 2019. Interpretation of neural networks is fragile. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, 3681–3688.
- Ghorbani, A.; Wexler, J.; Zou, J.; and Kim, B. 2019. Towards automatic concept-based explanations. *arXiv preprint arXiv:1902.03129*.



- Goguen, J. 2005. What is a concept? In *International Conference on Conceptual Structures*, 52–77. Springer.
- Goguen, J. A.; and Burstall, R. M. 1992. Institutions: Abstract model theory for specification and programming. *Journal of the ACM (JACM)*, 39(1): 95–146.
- Guidotti, R.; Monreale, A.; Ruggieri, S.; Pedreschi, D.; Turini, F.; and Giannotti, F. 2018. Local rule-based explanations of black box decision systems. *arXiv preprint arXiv:1805.10820*.
- Hastie, T. J. 2017. Generalized additive models. In *Statistical models in S*, 249–307. Routledge.
- Hochreiter, S.; and Schmidhuber, J. 1997. Long short-term memory. *Neural computation*, 9(8): 1735–1780.
- Hopfield, J. J. 1982. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the national academy of sciences*, 79(8): 2554–2558.
- Jiménez-Luna, J.; Grisoni, F.; and Schneider, G. 2020. Drug discovery with explainable artificial intelligence. *Nature Machine Intelligence*, 2(10): 573–584.
- Joyal, A.; and Street, R. 1991. The geometry of tensor calculus, I. *Advances in mathematics*, 88(1): 55–112.
- Kahneman, D. 2011. *Thinking, fast and slow*. macmillan.
- Karasmanoglou, A.; Antonakakis, M.; and Zervakis, M. 2022. Heatmap-based Explanation of YOLOv5 Object Detection with Layer-wise Relevance Propagation. In *2022 IEEE International Conference on Imaging Systems and Techniques (IST)*, 1–6. IEEE.
- Katis, P.; Sabadini, N.; and Walters, R. F. C. 2002. Feedback, trace and fixed-point semantics. *RAIRO-Theor. Inf. Appl.*, 36(2): 181–194.
- Kaufmann, L. 1987. Clustering by means of medoids. In *Proc. Statistical Data Analysis Based on the L1 Norm Conference, Neuchatel, 1987*, 405–416.
- Kim, B.; Khanna, R.; and Koyejo, O. O. 2016. Examples are not enough, learn to criticize! criticism for interpretability. *Advances in neural information processing systems*, 29.
- Kingma, D. P.; and Ba, J. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.
- Koh, P. W.; Nguyen, T.; Tang, Y. S.; Musmann, S.; Pierson, E.; Kim, B.; and Liang, P. 2020. Concept bottleneck models. In *International Conference on Machine Learning*, 5338–5348. PMLR.
- Kulkarni, A.; Shivananda, A.; and Sharma, N. R. 2022. Explainable AI for Computer Vision. In *Computer Vision Projects with PyTorch*, 325–340. Springer.
- Letham, B.; Rudin, C.; McCormick, T. H.; Madigan, D.; et al. 2015. Interpretable classifiers using rules and bayesian analysis: Building a better stroke prediction model. *Annals of Applied Statistics*, 9(3): 1350–1371.
- Li, Y.; Zhou, J.; Verma, S.; and Chen, F. 2022. A survey of explainable graph neural networks: Taxonomy and evaluation metrics. *arXiv preprint arXiv:2207.12599*.
- Lo Piano, S. 2020. Ethical principles in machine learning and artificial intelligence: cases from the field and possible ways forward. *Humanities and Social Sciences Communications*, 7(1): 1–7.
- Lundberg, S.; and Lee, S.-I. 2017. A unified approach to interpreting model predictions. *arXiv preprint arXiv:1705.07874*.
- Mac Lane, S. 1978. *Categories for the Working Mathematician*. Graduate Texts in Mathematics. Springer New York.
- Manhaeve, R.; Dumancic, S.; Kimmig, A.; Demeester, T.; and De Raedt, L. 2018. Deepproblog: Neural probabilistic logic programming. *Advances in Neural Information Processing Systems*, 31.
- Marcus, G. 2020. The next decade in AI: four steps towards robust artificial intelligence. *arXiv preprint arXiv:2002.06177*.
- Miller, T. 2019. Explanation in artificial intelligence: Insights from the social sciences. *Artificial intelligence*, 267: 1–38.
- Minh, D.; Wang, H. X.; Li, Y. F.; and Nguyen, T. N. 2022. Explainable artificial intelligence: a comprehensive review. *Artificial Intelligence Review*, 55(5): 3503–3568.
- Molnar, C. 2020. *Interpretable machine learning*. Lulu.com.
- Nelder, J. A.; and Wedderburn, R. W. 1972. Generalized linear models. *Journal of the Royal Statistical Society: Series A (General)*, 135(3): 370–384.
- Ong, E.; and Veličković, P. 2022. Learnable Commutative Monoids for Graph Neural Networks. *arXiv preprint arXiv:2212.08541*.
- Palacio, S.; Lucieri, A.; Munir, M.; Ahmed, S.; Hees, J.; and Dengel, A. 2021. Xai handbook: Towards a unified framework for explainable ai. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 3766–3775.
- Petsiuk, V.; Das, A.; and Saenko, K. 2018. Rise: Randomized input sampling for explanation of black-box models. *arXiv preprint arXiv:1806.07421*.
- Prawitz, D. 2006. *Natural deduction: A proof-theoretical study*. Courier Dover Publications.
- Ribeiro, M. T.; Singh, S.; and Guestrin, C. 2016a. ” Why should i trust you?” Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, 1135–1144.
- Ribeiro, M. T.; Singh, S.; and Guestrin, C. 2016b. Model-agnostic interpretability of machine learning. *arXiv preprint arXiv:1606.05386*.
- Ribeiro, M. T.; Singh, S.; and Guestrin, C. 2018. Anchors: High-precision model-agnostic explanations. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32.
- Riley, M. 2018. Categories of optics. *arXiv preprint arXiv:1809.00738*.
- Rudin, C. 2019. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature Machine Intelligence*, 1(5): 206–215.
- Rumelhart, D. E.; Hinton, G. E.; and Williams, R. J. 1985. Learning internal representations by error propagation. Technical report, California Univ San Diego La Jolla Inst for Cognitive Science.

- Santosa, F.; and Symes, W. W. 1986. Linear inversion of band-limited reflection seismograms. *SIAM Journal on Scientific and Statistical Computing*, 7(4): 1307–1330.
- Schmidt, M.; and Lipson, H. 2009. Distilling free-form natural laws from experimental data. *science*, 324(5923): 81–85.
- Selinger, P. 2001. Control categories and duality: on the categorical semantics of the lambda-mu calculus. *Mathematical Structures in Computer Science*, 11: 207 – 260.
- Selvaraju, R. R.; Cogswell, M.; Das, A.; Vedantam, R.; Parikh, D.; and Batra, D. 2017. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE international conference on computer vision*, 618–626.
- Shiebler, D.; Gavranović, B.; and Wilson, P. 2021. Category theory in machine learning. *arXiv preprint arXiv:2106.07032*.
- Simonyan, K.; Vedaldi, A.; and Zisserman, A. 2013. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*.
- Sprunger, D.; and Katsumata, S. 2019. Differentiable Causal Computations via Delayed Trace. In *34th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2019, Vancouver, BC, Canada, June 24-27, 2019*, 1–12. IEEE.
- Stein, D.; and Staton, S. 2021. Compositional Semantics for Probabilistic Programs with Exact Conditioning. In *2021 36th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*, 1–13.
- Swan, J.; Nivel, E.; Kant, N.; Hedges, J.; Atkinson, T.; and Steunebrink, B. 2022. *A Compositional Framework*, 73–90. Cham: Springer International Publishing.
- Takeuti, G. 2013. *Proof theory*, volume 81. Courier Corporation.
- Tarski, A. 1944. The semantic conception of truth: and the foundations of semantics. *Philosophy and phenomenological research*, 4(3): 341–376.
- Tibshirani, R. 1996. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, 58(1): 267–288.
- Turi, D.; and Plotkin, G. D. 1997. Towards a mathematical operational semantics. *Proceedings of Twelfth Annual IEEE Symposium on Logic in Computer Science*, 280–291.
- Uustalu, T.; and Vene, V. 2005. The Essence of Dataflow Programming. In Yi, K., ed., *Programming Languages and Systems, Third Asian Symposium, APLAS 2005, Tsukuba, Japan, November 2-5, 2005, Proceedings*, volume 3780 of *Lecture Notes in Computer Science*, 2–18. Springer.
- Uustalu, T.; and Vene, V. 2008. Comonadic notions of computation. *Electronic Notes in Theoretical Computer Science*, 203(5): 263–284.
- Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, Ł.; and Polosukhin, I. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.
- Verhulst, P. F. 1845. Resherches mathematiques sur la loi d’accroissement de la population. *Nouveaux memoires de l’academie royale des sciences*, 18: 1–41.
- Wachter, S.; Mittelstadt, B.; and Russell, C. 2017. Counterfactual explanations without opening the black box: Automated decisions and the GDPR. *Harv. JL & Tech.*, 31: 841.
- Wei, P.; Lu, Z.; and Song, J. 2015. Variable importance analysis: a comprehensive review. *Reliability Engineering & System Safety*, 142: 399–432.
- Wilson, P.; and Zanasi, F. 2021. Reverse Derivative Ascent: A Categorical Approach to Learning Boolean Circuits. In *CoRR*, volume 333, 247–260. Electronic Proceedings in Theoretical Computer Science.
- Yang, H.; Rudin, C.; and Seltzer, M. 2017. Scalable Bayesian rule lists. In *International conference on machine learning*, 3921–3930. PMLR.
- Zeiler, M. D.; and Fergus, R. 2014. Visualizing and understanding convolutional networks. In *European conference on computer vision*, 818–833. Springer.
- Zintgraf, L. M.; Cohen, T. S.; Adel, T.; and Welling, M. 2017. Visualizing deep neural network decisions: Prediction difference analysis. *arXiv preprint arXiv:1702.04595*.

## A Elements of Category Theory

### A.1 Monoidal Categories

The process interpretation of monoidal categories (Coecke and Kissinger 2017; Fritz 2020) sees morphisms in monoidal categories as modelling processes with multiple inputs and multiple outputs. Monoidal categories also provide an intuitive syntax for them through string diagrams (Joyal and Street 1991). The coherence theorem for monoidal categories (Mac Lane 1978) ensures that string diagrams are a sound and complete syntax for them and thus all coherence equations for monoidal categories correspond to continuous deformations of string diagrams. One of the main advantages of string diagrams is that they make reasoning with equational theories more intuitive.

**Definition A.1** (Eilenberg and MacLane (1945)). A *category*  $\mathcal{C}$  is given by a class of *objects*  $\mathcal{C}^o$  and, for every two objects  $X, Y \in \mathcal{C}^o$ , a set of *morphisms*  $\text{hom}(X, Y)$  with input type  $X$  and output type  $Y$ . A morphism  $f \in \text{hom}(X, Y)$  is written  $f: X \rightarrow Y$ . For all morphisms  $f: X \rightarrow Y$  and morphisms  $g: Y \rightarrow Z$  there is a *composite* morphisms  $f; g: X \rightarrow Z$ . For each object  $X \in \mathcal{C}^o$  there is an *identity* morphism  $\mathbb{1}_X \in \text{hom}(X, X)$ , which represents the process that “does nothing” to the input and just returns it as it is. Composition needs to be associative, i.e. there is no ambiguity in writing  $f; g; h$ , and unital, i.e.  $f; \mathbb{1}_Y = f = \mathbb{1}_X; f$  (Figure 3, bottom).

$$\begin{array}{c}
 X \text{ --- } \boxed{f} \text{ --- } \boxed{g} \text{ --- } Z \qquad X \text{ --- } X \\
 X \text{ --- } \boxed{f} \text{ --- } Y = X \text{ --- } \boxed{f} \text{ --- } Y = X \text{ --- } \boxed{f} \text{ --- } Y
 \end{array}$$

Figure 3: String diagrams for the composition of two morphisms (top, left), the identity morphism (top, right), and the unitality condition (bottom).

Monoidal categories (Mac Lane 1978) are categories endowed with extra structure, a monoidal product and a monoidal unit, that allows morphisms to be composed *in parallel*. The monoidal product is a functor  $\times: \mathcal{C} \times \mathcal{C} \rightarrow \mathcal{C}$  that associates to two processes,  $f_1: X_1 \rightarrow Y_1$  and  $f_2: X_2 \rightarrow Y_2$ , their parallel composition  $f_1 \times f_2: X_1 \times X_2 \rightarrow Y_1 \times Y_2$  (Figure 4, center). The monoidal unit is an object  $e \in \mathcal{C}^o$ , which represents the “absence of inputs or outputs” and needs to satisfy  $X \times e \cong X \cong e \times X$ , for each  $X \in \mathcal{C}^o$ . For this reason, this object is often not drawn in string diagrams and a morphism  $s: e \rightarrow Y$ , or  $t: X \rightarrow e$ , is represented as a box with no inputs, or no outputs.

$$\boxed{s} \text{ --- } Y \qquad \text{and} \qquad X \text{ --- } \boxed{t}$$

### A.2 Cartesian and Symmetric Monoidal Categories

A symmetric monoidal structure on a category is required to satisfy some coherence conditions (Mac Lane 1978), which ensure that string diagrams are a sound and complete syntax for symmetric monoidal categories (Joyal and Street 1991). Like functors are mappings between categories that preserve their structure, *symmetric monoidal functors* are mappings between symmetric monoidal categories that preserve the structure and axioms of symmetric monoidal categories.

A monoidal category is *symmetric* if there is a morphism  $\sigma_{X,Y}: X \times Y \rightarrow Y \times X$ , for any two objects  $X$  and  $Y$ , called the *symmetry*. The inverse of  $\sigma_{X,Y}$  is  $\sigma_{Y,X}$ :

$$\begin{array}{c}
 X_1 \\
 X_2 \\
 X_3
 \end{array}
 \text{ --- } \boxed{h} \text{ --- }
 \begin{array}{c}
 Y_1 \\
 Y_2
 \end{array}
 \qquad
 \begin{array}{c}
 X_1 \text{ --- } \boxed{f_1} \text{ --- } Y_1 \\
 X_2 \text{ --- } \boxed{f_2} \text{ --- } Y_2
 \end{array}
 \qquad
 \begin{array}{c}
 X \\
 Y
 \end{array}
 \begin{array}{c}
 \diagup \\
 \diagdown
 \end{array}
 \begin{array}{c}
 Y \\
 X
 \end{array}$$

Figure 4: A morphism with multiple inputs and outputs (left), the parallel composition of two morphisms (center), and the symmetry (right) in a monoidal category.

$$\begin{array}{c}
 X \\
 Y
 \end{array}
 \begin{array}{c}
 \diagup \\
 \diagdown
 \end{array}
 \begin{array}{c}
 X \\
 Y
 \end{array}
 =
 \begin{array}{c}
 X \text{ --- } X \\
 Y \text{ --- } Y
 \end{array}$$

These morphisms need to be coherent with the monoidal structure, e.g.

$$\begin{array}{c}
 X \times Y \\
 Z
 \end{array}
 \begin{array}{c}
 \diagup \\
 \diagdown
 \end{array}
 \begin{array}{c}
 Z \\
 X \times Y
 \end{array}
 =
 \begin{array}{c}
 X \text{ --- } Z \\
 Y \text{ --- } X \\
 Z \text{ --- } Y
 \end{array}
 .$$

Moreover, the symmetries are a natural transformation, i.e. morphisms can be swapped,

$$\begin{array}{c}
 X \text{---} \boxed{f} \\
 Z \text{---} \boxed{g}
 \end{array}
 \begin{array}{c}
 \diagup \\
 \diagdown
 \end{array}
 \begin{array}{c}
 W \\
 Y
 \end{array}
 =
 \begin{array}{c}
 X \text{---} \boxed{g} \\
 Z \text{---} \boxed{f}
 \end{array}
 \begin{array}{c}
 \diagdown \\
 \diagup
 \end{array}
 \begin{array}{c}
 W \\
 Y
 \end{array}
 .$$

Some symmetric monoidal categories have additional structure that allows resources to be copied and discarded (Fox 1976). These are called *Cartesian categories*. It is customary to indicate with  $\times$  the monoidal product given by the cartesian structure and with  $e$  the corresponding monoidal unit. Cartesian categories are equipped, for every object  $X$ , with morphisms  $\nu_X : X \rightarrow X \times X$  and  $\epsilon_X : X \rightarrow e$ :

$$X \text{---} \bullet \begin{array}{c} \diagup \\ \diagdown \end{array} \begin{array}{c} X \\ X \end{array} \quad \text{and} \quad X \text{---} \bullet$$

These need to be natural transformations, i.e. morphisms can be copied and discarded,

$$\begin{array}{c}
 X \text{---} \boxed{f} \\
 \bullet
 \end{array}
 \begin{array}{c}
 \diagup \\
 \diagdown
 \end{array}
 \begin{array}{c}
 Y \\
 Y
 \end{array}
 =
 \begin{array}{c}
 X \text{---} \bullet \\
 \begin{array}{c} \boxed{f} \\ \boxed{f} \end{array}
 \end{array}
 \begin{array}{c}
 \diagup \\
 \diagdown
 \end{array}
 \begin{array}{c}
 Y \\
 Y
 \end{array}$$

$$X \text{---} \boxed{f} \text{---} \bullet = Y \text{---} \bullet$$

be coherent with the monoidal structure

$$\begin{array}{c}
 X \times Y \text{---} \bullet \\
 \begin{array}{c} X \times Y \\ X \times Y \end{array}
 \end{array}
 =
 \begin{array}{c}
 X \text{---} \bullet \\
 Y \text{---} \bullet \\
 \begin{array}{c} \diagup \\ \diagdown \end{array} \\
 \begin{array}{c} X \\ Y \end{array}
 \end{array}$$

$$X \times Y \text{---} \bullet = \begin{array}{c} X \text{---} \bullet \\ Y \text{---} \bullet \end{array}$$

and satisfy the equations of a cocommutative comonoid.

$$\begin{array}{c}
 X \text{---} \bullet \\
 \begin{array}{c} \diagup \\ \diagdown \end{array} \\
 \begin{array}{c} X \\ X \\ X \end{array}
 \end{array}
 =
 \begin{array}{c}
 X \text{---} \bullet \\
 \begin{array}{c} \diagdown \\ \diagup \end{array} \\
 \begin{array}{c} X \\ X \\ X \end{array}
 \end{array}$$

$$\begin{array}{c}
 X \text{---} \bullet \\
 \begin{array}{c} \diagup \\ \diagdown \end{array} \\
 \begin{array}{c} X \\ X \end{array}
 \end{array}
 =
 X \text{---} X$$

$$\begin{array}{c}
 X \text{---} \bullet \\
 \begin{array}{c} \diagup \\ \diagdown \end{array} \\
 \begin{array}{c} X \\ X \end{array}
 \end{array}
 =
 \begin{array}{c}
 X \text{---} \bullet \\
 \begin{array}{c} \diagdown \\ \diagup \end{array} \\
 \begin{array}{c} X \\ X \end{array}
 \end{array}$$

### A.3 Feedback Monoidal Categories

**Definition A.2.** A *feedback monoidal category* is a symmetric monoidal category  $\mathcal{C}$  endowed with an endofunctor  $F : \mathcal{C} \rightarrow \mathcal{C}$ , and an operation  $\circ_S : \text{hom}(X \times F(S), Y \times S) \rightarrow \text{hom}(X, Y)$  for all objects  $X, Y, S$  in  $\mathcal{C}$ , which satisfies the following axioms:

- (Tightening)  $\circ_S ((g \times \mathbb{1}_{FS}); f; (h \times \mathbb{1}_S)) = g; \circ_S (f); h;$
- (Joining)  $\circ_{S \times T} (f) = \circ_S (\circ_T (f));$
- (Vanishing)  $\circ_1 (f) = f;$
- (Strength)  $\circ_S (g \times f) = g \times \circ_S (f);$
- (Sliding)  $\circ_T (f; (\mathbb{1}_Y \times g)) = \circ_S ((\mathbb{1}_X \times g); f).$

*Feedback monoidal functors* are mappings between feedback monoidal categories that preserve the structure and axioms of feedback monoidal categories.

Feedback monoidal categories are the *syntax* for processes with feedback loops. When the monoidal structure of a feedback monoidal category is cartesian, we call it *feedback cartesian category*. Their *semantics* can be given by monoidal streams (Di Lavore, de Felice, and Román 2022). In cartesian categories, these have an explicit description. We refer to them as cartesian streams, but they have appeared in the literature multiple times under the name of “stateful morphism sequences” (Sprunger and Katsumata 2019) and “causal stream functions” (Uustalu and Vene 2005).

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

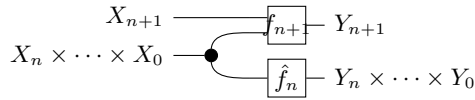
Table 1: General learning schemes. In the table the input, output are supposed to be arbitrary but of fixed type (i.e.  $\diamond^{\mathbb{N}}$ ). The architecture is supposed to be potentially variable over the time (i.e.  $\ast$ ). To express a fixed architecture with fixed domain is sufficient to add the constraint  $\mathcal{T}(\eta)_i = \mathcal{T}(\eta)_{i+1}$ .

#### A.4 Cartesian Streams

A *cartesian stream*  $\mathbb{f}: \mathbb{X} \rightarrow \mathbb{Y}$ , with  $\mathbb{X} = (X_0, X_1, \dots)$  and  $\mathbb{Y} = (Y_0, Y_1, \dots)$ , is a family of functions  $f_n: X_n \times \dots \times X_0 \rightarrow Y_n$  indexed by natural numbers. Cartesian streams form a category  $\text{Streams}_{\text{Set}}$ .

Each  $f_n$  gives the output of the stream at time  $n$ . We can compute the outputs until time  $n$  by combining  $f_0, \dots, f_n$  to get  $\hat{f}_n: X_n \times \dots \times X_0 \rightarrow Y_n \times \dots \times Y_0$  as follows:

- $\hat{f}_0 := f_0$
- $\hat{f}_{n+1} := (\mathbb{1}_{X_{n+1}} \times \nu_{X_n \times \dots \times X_0}); (f_{n+1} \times \hat{f}_n)$



The composition of two cartesian streams  $\mathbb{f}: \mathbb{X} \rightarrow \mathbb{Y}$  and  $\mathbb{g}: \mathbb{Y} \rightarrow \mathbb{Z}$  has components  $(\mathbb{f}; \mathbb{g})_n := \hat{f}_n; g_n$ , and the identity stream has projections as components  $(\mathbb{1}_{\mathbb{X}})_n := \pi_{X_n}$ .

#### A.5 Free Categories

We generate “abstract” categories using the notion of *free category* (Mac Lane 1978). Intuitively, a free category serves as a template for a class of categories (e.g., feedback monoids). To generate a free category, we just need to specify a set of objects and morphisms generators. Then we can realize “concrete” instances of a free category  $F$  using a functor from  $F$  to another category  $C$  that preserves the axioms of  $F$ . If such a functor exists then  $C$  is of the same type of  $F$  (e.g., the image of a free feedback monoidal category via a feedback functor is a feedback monoidal category).

#### A.6 Institutions

An *institution*  $I$  is constituted by:

- a category  $\text{Sign}_I$  whose objects are signatures (i.e. vocabularies of symbols);
- a functor  $\text{Sen}: \text{Sign}_I \mapsto \text{Set}$  providing sets of well-formed expressions ( $\Sigma$ -sentences) for each signature  $\Sigma \in \text{Sign}_I$ ;
- a functor  $\text{Mod}: \text{Sign}_I^{\text{op}} \mapsto \text{Set}$  providing semantic interpretations, i.e. worlds.

Furthermore, Satisfaction is then a parametrized relation  $\models_{\Sigma}$  between  $\text{Mod}(\Sigma)$  and  $\text{Sen}(\Sigma)$ , such that for all signature morphism  $\rho: \Sigma \mapsto \Sigma'$ ,  $\Sigma'$ -model  $M'$ , and any  $\Sigma$ -sentence  $e$ ,

$$M' \models_{\Sigma} \rho(e) \text{ iff } \rho(M') \models_{\Sigma} e$$

where  $\rho(e)$  abbreviates  $\text{Sen}(\rho)(e)$  and  $\rho(M')$  stands for  $\text{Mod}(\rho)(e)$ .

#### A.7 Expressive power of abstract learning agents

Abstract explaining learning agents have the capability of encode a wide range of known learning processes given a suitable translator functor. Indeed, Definition 3.5 allows to encode any learning scheme with an input ( $X$ ), an optimized output ( $Y$ ), a supervision ( $Y^*$ ), and an unoptimized output ( $E$ ) of any conceivable type. It is important to note that there is no restriction in how  $X, Y, Y^*, E$  can be instantiated by a translator functor since, as object in a free category, they act as empty boxes concretely filled by the given translator. The following Tables 1 and 2 we show the fundamental characteristics of translator functors instantiating various type of learning processes with or without explanations as output of the model. We will use the symbol  $\diamond$  as wildcard of dataspace meaning any type of data and  $\ast$  as a wildcard for the type of functions and we denote with  $\mathcal{T}'$  an additional generic LA.

Learning models	$\mathcal{T}(X)$	$\mathcal{T}(Y)$	$\mathcal{T}(Y^*)$	$\mathcal{T}(E)$	$\mathcal{T}(\eta)$
MLPs	$\diamond^{\mathbb{N}}$	$\diamond^{\mathbb{N}}$	$\diamond^{\mathbb{N}}$	$\{*\}^{\mathbb{N}}$	$\mathcal{T}(\eta)_i = \mathcal{T}(\eta)_{i+1}$
RNNs	$X^{\mathbb{N}} \times H^{\mathbb{N}}$	$Y^{\mathbb{N}} \times H^{\mathbb{N}}$	$Y^{\mathbb{N}}$	$\{*\}^{\mathbb{N}}$	$\mathcal{T}(\eta)_i = \mathcal{T}(\eta)_{i+1}$
nural arch. search	$\diamond^{\mathbb{N}}$	$\diamond^{\mathbb{N}}$	$\mathcal{T}(Y)$	$\{*\}^{\mathbb{N}}$	$\star$
CBMs	$\diamond^{\mathbb{N}}$	$\diamond^{\mathbb{N}}$	$\mathcal{T}(Y)$	$\diamond^{\mathbb{N}}$	$(\hat{\mu} \times \mathbb{1}_P); \hat{\eta}'$
Post-hoc exps	$\mathcal{T}'(Y) \times \mathcal{T}'(X) \times \mathcal{T}'(P)$	$\diamond^{\mathbb{N}}$	$\mathcal{T}(Y)$	$\diamond^{\mathbb{N}}$	$\mathcal{T}(\eta)_i = \mathcal{T}(\eta)_{i+1}$
Intrin. exps	$\diamond^{\mathbb{N}}$	$\diamond^{\mathbb{N}}$	$\mathcal{T}(Y)$	$\diamond^{\mathbb{N}}$	$\mathcal{T}(\eta)_i = \mathcal{T}(\eta)_{i+1}$
Model-agn. exps	$\mathcal{T}'(Y) \times \mathcal{T}'(X)$	$\diamond^{\mathbb{N}}$	$\mathcal{T}(Y)$	$\diamond^{\mathbb{N}}$	$\mathcal{T}(\eta)_i = \mathcal{T}(\eta)_{i+1}$
Model-spc. exps	$\mathcal{T}'(Y) \times \mathcal{T}'(X) \times \mathcal{T}'(P)$	$\diamond^{\mathbb{N}}$	$\mathcal{T}(Y)$	$\diamond^{\mathbb{N}}$	$\mathcal{T}(\eta)_i = \mathcal{T}(\eta)_{i+1}$
Backw.-base exps	$\diamond^{\mathbb{N}} \times \mathcal{T}'(P)$	$\diamond^{\mathbb{N}}$	$\mathcal{T}(Y)$	$\diamond^{\mathbb{N}}$	$\mathcal{T}(\eta)_i = \mathcal{T}(\eta)_{i+1}$
Forw.-base exps	$\diamond^{\mathbb{N}} \times h(\mathcal{T}(P))$	$\diamond^{\mathbb{N}}$	$\mathcal{T}(Y)$	$\diamond^{\mathbb{N}}$	$\mathcal{T}(\eta)_i = \mathcal{T}(\eta)_{i+1}$

Table 2: Standard learning models. In the table  $H$  is the space of the state vector. For the explainers,  $\mathcal{T}'$  is a translator instantiating an LA to be explained,  $h(\mathcal{P}) = \frac{\partial \mathcal{L}(\mathcal{Y}, \mathcal{Y})}{\partial \mathcal{P}}$  is the gradient of the loss function  $\mathcal{L}$  on  $\mathcal{P}$ , and  $\mathcal{T}(\eta)_i = \mathcal{T}(\eta)_{i+1}$  represents a fixed architecture with fixed domain.